

明 細 書

グラフ幅削減装置及びグラフ幅削減方法、並びに論理回路合成装置及び論理回路合成方法

技術分野

[0001] 本発明は、ルックアップ・テーブル(以下、「LUT」という。)-カスケード論理回路やLUT型FPGA等において使用される、多出力論理関数 $f(X)$ に対応する論理回路を構成するためのデータであるLUTを生成するための論理回路合成技術に関するものである。

背景技術

[0002] 近年では、種々の電子回路の設計において、LUT型のフィールド・プログラマブル・ゲートアレイ(Field Programmable Gate Array : 以下、「FPGA」という。)が広く使用されている(例えば、非特許文献1〜3参照)。LUT型FPGAは、多数の再構成可能論理ブロック(Configurable Logic Block : 以下、「CLB」という。)が格子状に配列され、各CLBの間に縦横に格子状に配線された複数の接続線(routing line : 繞線)を備えた構成からなる。各々の接続線の交点には、再構成可能なスイッチ・ブロック(Switch Block : 以下「SB」という。)が配されており、各々の接続線の接続を自由に変更(再構成)することができる。また、各CLBと接続線とは、再構成可能な接続ブロック(Connection Block : 以下、「CB」という。)により接続されている。そして、各接続線の終端には、FPGA外部との入出力を行う入出力部(I/O)が設けられている。各CLBは、内部に多入力1出力のLUTやマルチプレクサ(MUX)を1ないし複数個有し、このLUTやMUXにより論理演算が可能である。LUT型FPGAでは、SB、CB及びCLBの内部を再構成することにより、所望の多入力1出力論理関数が格納された複数のLUTの入出力を目的に応じてネットワーク状に順序接続し、様々な組み合わせ論理回路を実現することができる。

[0003] 一方、FPGAよりも高速な再構成可能論理回路を実現するものとして、LUTカスケード論理回路が提案されている(例えば、非特許文献4, 5参照)。LUTカスケード論理回路においては、多入力多出力のLUTがカスケード状に接続された構成からなる

。各LUTには、外部からの入力に加えて前段のLUTの出力が入力される。そして、最終段のLUTから1ないし複数の出力変数が出力される。演算を行おうとしている論理関数を複数の多出力論理関数に分解し、各多出力論理関数は各段のLUTに格納される。このようにして、LUTカスケード論理回路においては多出力論理関数の演算を行うことができる。

- [0004] 上記のようなLUT型FPGAやLUTカスケード論理回路を、実際の論理回路設計に適用する場合には、まず、論理回路で実現しようとする論理関数(以下、「目的論理関数」という。)を複数の論理関数に関数分解して合成論理関数とする。ここで、合成論理関数とは、関数分解により得られる複数の論理関数(以下、「分解関数」という。))の結合論理からなる関数をいう。すなわち、目的論理関数を $f(X)$ (X は入力変数の集合)で表した場合に、この目的論理関数を、 $f(X_1, X_2) = g(h(X_1), X_2)$ ($\{X_1\} \subset X$, $\{X_2\} \subset X$, $\{X_1\} \cap \{X_2\} = \phi$, $\{X_1\} \cup \{X_2\} = X$)の形に関数分解した場合、 $g(h(X_1), X_2)$ を g と h の合成論理関数という。
- [0005] 尚、本明細書において、 X, Y と記す場合には、入力変数、出力変数のベクトル又は全順序集合(ordered set)を表し、 $\{X\}, \{Y\}$ と記すときは、入力変数、出力変数の非順序集合(unordered set)を表すものとする。
- [0006] 上記関数分解により、2つの分解関数 $h(X_1)$, $g(h, X_2)$ が得られる。尚、かかる関数分解は常に可能であるとは限らないが、コンピュータ等で使用する制御回路や算術演算回路等の多くの実用的関数は分解可能なものが多い(非特許文献6参照)。また、分解関数 $g(h, X_2)$ が更に分解可能であれば同様の関数分解を繰り返す。
- [0007] そして、各分解関数を別々のLUTとして実現し、各LUTを合成論理関数に従ってネットワーク状又はカスケード状に接続する。このようにして、目的論理関数をLUT型FPGAやLUTカスケード論理回路によって実現することができる。
- [0008] 単一出力の目的論理関数を、上記関数分解の手法を用いて、LUTが結合した論理回路により実現することは比較的容易である(例えば、非特許文献4, 5参照)。
- [0009] 一方、目的論理関数が多出力である場合において、目的論理関数をLUTが結合した論理回路により実現する論理合成技術としては、現在のところ以下の方法によるものが知られている。

(1) 多端子二分決定グラフ (Multi-Terminal Binary Decision Diagram: MTBDD) を用いる方法 (非特許文献7, 8参照)

(2) 出力を幾つかのグループに分割して構成する方法 (非特許文献7参照)

(3) 分割理論を用いる方法 (非特許文献9, 10参照)

(4) 代入による方法 (非特許文献11参照)

(5) Hyper Functionを用いる方法 (非特許文献12参照)

(6) 非零出力符号化特性関数 (Encoded Characteristic Function for Non-zero : ECFN) を用いた時分割実現による方法 (非特許文献4, 5参照)

(7) 以上のうちいくつかの方法の組み合わせによる方法 (非特許文献11参照)

非特許文献1: S. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic,

"Field-Programmable Gate Arrays", Kluwer Academic Publishers, 1992.

非特許文献2: P. Chow, S. O. Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja,

"The design of an SRAM-based field-programmable gate array —Part I:

Architecture," IEEE Transactions on VLSI Systems, vol.7, pp.191-197, June 1999.

非特許文献3: Chow, P., S. Seo, J. Rose, K. Chung, G. Pez-Monzon and I. Rahardja.

"The Design of an SRAM-Based Field-Programmable Gate Array, Part II: Circuit

Design and Layout", IEEE Transactions on Very Large Scale Integration (VLSI)

Systems, Vol. 7 No. 3, Sept. 1999, pp. 321-330.

非特許文献4: 笹尾勤, 松浦宗寛, 井口幸洋, "多出力関数のカスケード実現と再構成可能ハードウェアによる実現", 電子情報通信学会FTS研究会, FTS2001-8, pp.

57-64, 三重大学(2001-04).

非特許文献5: T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," International Workshop on

Logic and Synthesis (IWLS01), Lake Tahoe, CA, June 12-15, 2001. pp.225-230.

非特許文献6: T. Sasao and M. Matsuura, "DECOMPOS: An integrated system for functional decomposition", 1998 International Workshop on Logic Synthesis, Lake

Tahoe, June 1998.

非特許文献7: Y-T. Lai, M. Pedram and S. B. K. Vrudhula, "BDD based decomposition

of logic functions with application to FPGA synthesis", 30th ACM/IEEE Design Automation Conference, June 1993.

非特許文献8:T.Sasao, "FPGA design by generalized functional decomposition", In Logic Synthesis and Optimization, Sasao ed., Kluwer Academic Publisher, pp.233-258, 1993.

非特許文献9:C.Scholl and P.Molitor, "Communication based FPGA synthesis for multi-output Boolean functions", Asia and South Pacific Design Automation Conference, pp.279-287, Aug. 1995.

非特許文献10:B.Wurth, K.Eckl, and K.Anterich, "Functional multiple-output decomposition: Theory and implicit algorithm", Design Automation Conf., pp.54-59, June 1995.

非特許文献11:H.Sawada, T.Suyama, and A.Nagoya, "Logic synthesis for look-up table based FPGAs using functional decomposition and support minimization", ICCAD, pp.353-359, Nov. 1995.

非特許文献12:J.-H.R.Jian, J.-Y.Jou, and J.-D.Huang, "Compatible class encoding in hyper-function decomposition for FPGA synthesis", Design Automation Conference, pp.712-717, June 1998.

非特許文献13:P.Ashar and S.Malik, "Fast functional simulation using branching programs", Proc. International Conference on Computer Aided Design, pp.408-412, Nov. 1995.

非特許文献14:C.Scholl, R.Drechsler, and B.Becker, "Functional simulation using binary decision diagram", ICCAD'97, pp.8-12, Nov. 1997.

非特許文献15:A.Mishchenko and T.Sasao, "Logic Synthesis of LUT Cascades with Limited Rails", 電子情報通信学会VLSI設計技術研究会, VLD2002-9, 琵琶湖 (2002-11)

非特許文献16:M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman & Co., New York, 1979.

発明の開示

発明が解決しようとする課題

- [0010] 上記背景技術に挙げた(1)～(7)の方法による論理合成技術は、何れも多出力論理関数の論理合成の技術としては決定的な方法であるとはいえない。例えば、MTBDDを用いて多出力論理関数の関数分解を行うことは、理論上は可能である。しかしながら、 m 出力の論理関数の場合、最大で 2^m 個の終端節点が存在する。従って、MTBDDを用いて多出力論理関数の関数分解を行った場合には、多くの場合、列複雑度(後述の[定義6]参照)が増大し、実用的ではない。また、他の従来の方法に関しても同様なことがいえる。
- [0011] 一方、比較器や加算器のように、すべての論理演算が終了しない段階においても、演算を進めるに従って、順次出力変数が決定していくような論理関数も多い。例えば、入力変数 $P=(p_1, p_2, p_3, p_4)$ ($P \in B^4$, $B=\{0, 1\}$) 及び $Q=(q_1, q_2, q_3, q_4)$ ($Q \in B^4$) を加算して出力変数 $R=(r_0, r_1, r_2, r_3, \text{cout})$ ($R \in B^5$) を出力するような加算器では、各桁の加算が終了した時点で、すべての桁の演算が終了する前にその桁に対応する出力変数が決定する。このように、すべての論理演算が終了しない段階で決定し出力される出力変数を「中間出力変数」という。
- [0012] 目的論理関数が中間出力変数を有する場合、目的論理関数を実現するLUTネットワークやLUTカスケードにおいては、最終段以外の段のLUTにおいて中間出力変数を出力するように構成することにより、各LUTのサイズを小さくするとともに演算速度を向上させることが可能となる。
- [0013] しかしながら、上記背景技術に挙げた論理合成技術では、中間出力変数を有する目的論理関数に対して、中間出力を有するLUTネットワークやLUTカスケードを合成することは困難であった。
- [0014] また、基数変換回路や加算回路などにおいては、各出力値は必ずしもすべての入力に対して定義されておらず、一部の入力変数に対しては出力値が0でも1でもよいようなドント・ケアを含む場合が多い。このようなドント・ケアを含む多出力論理関数をそのままLUTネットワークやLUTカスケードとして合成すると、メモリの冗長性が大きくなる。従って、ドント・ケアを含む多出力論理関数についてはできるだけLUTネットワークやLUTカスケードの冗長性が少なくなる形式に変更することが好ましい。

[0015] そこで、本発明の第1の目的は、一般的な多出力論理関数に対して汎用的に論理合成が可能であり、また、中間出力を有するLUTネットワークやLUTカスケードを合成することが可能な論理回路合成技術を提供することにある。

[0016] また、本発明の第2の目的は、ドント・ケアを含む多出力論理関数を、LUTネットワークやLUTカスケードの冗長性が少なくなる形式に変更する一手法であるグラフ幅削減技術を提供することにある。

課題を解決するための手段

[0017] 以下の説明では、まず本明細書で使用する主要な用語について定義し、本発明の基本的原理について説明した後、本発明の構成及び作用について説明する。

[0018] [1]用語の定義等

[定義1] (サポート変数)

関数 F が変数 x に依存するとき、 x を F の「サポート変数 (support variable)」という。関数 F のサポート変数の集合を $\text{supp}(F)$ と記す。

[定義終わり]

[0019] [定義2] (完全定義関数)

ブール変数を x と記し、 x 全体の集合を $B=\{0,1\}$ と記す。論理変数 $X (\in B^n, n$ は任意の自然数)の関数 $F(X) : B^n \rightarrow B^m$ (m は任意の自然数)を「完全定義関数 (completely specified function)」という。特に、 m が2以上の自然数の場合を「完全定義多出力関数 (completely specified multiple-output function)」といい、 $m=1$ の場合を「完全定義単一出力関数 (completely specified single-output function)」という。

[定義終わり]

[0020] [定義3] (不完全定義関数)

ある関数 $f(X)$ の関数値が0でも1でもよいことを「ドント・ケア (don't care)」といい、 d と記す。論理変数 $X (\in B^n, n$ は任意の自然数)の関数 $F(X) : B^n \rightarrow \{0,1,d\}^m$ (m は任意の自然数)を「不完全定義関数 (incompletely specified function)」という。特に、 m が2以上の自然数の場合を「不完全定義多出力関数 (incompletely specified multiple-output function)」といい、 $m=1$ の場合を「不完全定義単一出力関数 (incompletely specified single-output function)」という。

[定義終わり]

[0021] [定義4] (変数の分割)

論理関数の入力変数を $X=(x_1, x_2, \dots, x_n)$ とする。 X の変数の集合を $\{X\}$ で表す。 $\{X_1\} \cup \{X_2\} = \{X\}$ かつ $\{X_1\} \cap \{X_2\} = \phi$ のとき、 $X=(X_1, X_2)$ を「 X の分割(partition)」という。また、 X の変数の個数を $|X|$ で表す。

[定義終わり]

[0022] [定義5] (分解表)

論理関数 $F(X)$ に対して、 X の分割を $X=(X_1, X_2)$ とする。また、 $|X_1|=n_1$ 、 $|X_2|=n_2$ とする。このとき、 2^{n_1} 列 2^{n_2} 行の表で、各行、各列に2進符号のラベルを持ち、その要素が F の真理値であるような表を、 F の「分解表(decomposition chart)」という。このとき、 X_1 を「束縛変数(bind variable)」、 X_2 を「自由変数(free variable)」という。ここで、分解表の列、行は、それぞれ n_1 、 n_2 ビットのすべてのパターンを有する。

[定義終わり]

[0023] [定義6] (列複雑度、列関数)

分解表の異なる列パターンの数を「列複雑度(column multiplicity)」といい、 μ と記す。列パターンが表す関数を「列関数(column function)」という。

[定義終わり]

[0024] [例1]

(表1)に4入力1出力不完全定義関数の分解表を示す。すべての列パターンが異なるので、列複雑度 μ は4となる。

[0025] [表1]

		$X_1 = \{x_1, x_2\}$			
		00	01	10	11
$X_2 = \{x_3, x_4\}$	00	0	0	d	1
	01	1	1	d	d
	10	d	1	0	d
	11	0	d	0	0
		Φ_1	Φ_2	Φ_3	Φ_4

[例終わり]

[0026] [定義7] (ON関数、OFF関数、DC関数)

$f(X)=0, 1, d$ を満たすすべての X の集合を、それぞれ、「OFF集合 (OFF-set)」、「ON集合 (ON-set)」、「DC集合 (DC-set)」といい、 $f^{-1}(0)$, $f^{-1}(1)$, $f^{-1}(d)$ と記す。論理変数 $X (\in B^n)$ の不完全定義関数を $f(X)$ とする。OFF集合 $f^{-1}(0)$ に属するすべての X に1を対応づけ、それ以外の X に0を対応づける関数を「OFF関数 (OFF-function)」といい、 f_{-0} と記す。

[0027] [数1]

$$f_{-0}(X) = \begin{cases} 1 & (X \in f^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases} \quad (1)$$

[0028] ON集合 $f^{-1}(1)$ に属するすべての X に1を対応づけ、それ以外の X に0を対応づける関数を「ON関数 (ON-function)」といい、 f_{-1} と記す。

[0029] [数2]

$$f_{-1}(X) = \begin{cases} 1 & (X \in f^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases} \quad (2)$$

[0030] DC集合 $f^{-1}(d)$ に属するすべての X に1を対応づけ、それ以外の X に0を対応づける関数を「DC関数 (DC-function)」といい、 f_{-d} と記す。

[0031] [数3]

$$f_{-d}(X) = \begin{cases} 1 & (X \in f^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (3)$$

[0032] $f(X)$ が完全定義関数の場合にも、OFF関数 f_{-0} 、ON関数 f_{-1} を同様に定義する。特に $f(X)$ が完全定義単一出力関数の場合、

[0033] [数4]

$$f_{-0} = \bar{f}(X); \quad f_{-1} = f(X) \quad (4)$$

である。

[定義終わり]

[0034] [定義8] (完全定義関数の特性関数)

入力変数を $X=(x_1, x_2, \dots, x_n) (\in B^n)$ とし、完全定義多出力関数を $F(X)=(f_0(X), f_1(X), \dots, f_{m-1}(X))$ (m は自然数)とする。このとき、以下の式(5)のように定義される関数 $\chi(X, Y)$ を多出力論理関数 F の「特性関数(characteristic function)」という(非特許文献14参照)。ここで、 $Y=(y_0, \dots, y_{m-1})$ は出力を表す変数である。

[0035] [数5]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X)) \quad (5)$$

[定義終わり]

[0036] n 入力 m 出力関数の特性関数は $(n+m)$ 変数の二値論理関数であり、入力変数 x_i ($i=1, 2, \dots, n$)の他に、各出力 f_j ($j=0, 1, \dots, m-1$)に対して出力変数 y_j を用いる。 $B=\{0, 1\}$ とする。 $X \in B^n$ かつ $F(X)=(f_0(X), f_1(X), \dots, f_{m-1}(X)) \in B^m$ とする。いま、 $Y \in B^m$ とすると、論理関数 $F(X)$ の特性関数 $\chi(X, Y)$ の値は式(6)のようになる。

[0037] [数6]

$$\chi(X, Y) = \begin{cases} 1 & (Y = F(X)) \\ 0 & (Y \neq F(X)) \end{cases} \quad (6)$$

[0038] 完全定義多出力関数の特性関数は、入力の組み合わせと出力の組み合わせのうち許されているものを表す。式(7)のようにおくと、特性関数 χ は式(8)のように表現することができる。

[0039] [数7]

$$f_{i,0}(X) = \bar{f}_i(X), \quad f_{i,1}(X) = f_i(X) \quad (7)$$

[0040] [数8]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (\bar{y}_i \cdot f_{i,0} \vee y_i \cdot f_{i,1}) \quad (8)$$

[0041] 一方、不完全定義関数では、ドント・ケアの場合、その関数値が0でも1でもよい。従って、特性関数では、出力変数 y_i の値が0でも1でも真となる。そこで、ドント・ケアを表現する不定値関数 $f_{i,d}$ に対して、次式が成立する。

[0042] [数9]

$$\bar{y}_i(f_{i-0} \vee f_{i-d}) \vee y_i(f_{i-1} \vee f_{i-d}) = \bar{y}_i f_{i-0} \vee y_i f_{i-1} \vee f_{i-d} \quad (9)$$

[0043] 式(8)、式(9)より、完全定義多出力関数の特性関数を拡張して、不完全定義関数の特性関数 χ を次のように定義する。

[0044] [定義9] (不完全定義関数の特性関数)

入力変数 $X=(x_1, x_2, \dots, x_n) (\in B^n)$ の不完全定義多出力関数を $F(X)=(f_0(X), f_1(X), \dots, f_{m-1}(X))$ とする。このとき、以下の式(10)により定義される関数 $\chi(X, Y)$ を不完全定義多出力関数 F の特性関数という。ここで、 $Y=(y_0, \dots, y_{m-1})$ は出力を表す変数である。

[0045] [数10]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{\bar{y}_i f_{i-0} \vee y_i f_{i-1} \vee f_{i-d}\} \quad (10)$$

[定義終わり]

[0046] [例2]

(表2)に示した4入力2出力の不完全定義関数を考える。

[0047] [表2]

x_1	x_2	x_3	x_4	f_0	f_1
0	0	0	0	d	1
0	0	0	1	d	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	d	d
0	1	0	1	d	d
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	d
1	1	0	1	1	d
1	1	1	0	d	0
1	1	1	1	d	1

(表2)より、各OFF関数、ON関数、DC関数は次のように記述される。

[0048] [数11]

$$\begin{aligned}
 f_{0.0} &= \bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \\
 f_{0.1} &= \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \\
 f_{0.d} &= \bar{x}_1 \bar{x}_3 \vee x_1 x_2 x_3 \\
 f_{1.0} &= \bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_2 x_3 \bar{x}_4 \\
 f_{1.1} &= \bar{x}_2 \bar{x}_3 \vee x_2 x_3 x_4 \\
 f_{1.d} &= x_2 \bar{x}_3
 \end{aligned} \tag{11}$$

従って、この関数の特性関数は、次式のようになる。

[0049] [数12]

$$\begin{aligned}
 \chi &= \{ \bar{y}_0 (\bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3) \vee y_0 (\bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3) \vee (\bar{x}_1 \bar{x}_3 \vee x_1 x_2 x_3) \} \cdot \\
 &\quad \{ \bar{y}_1 (\bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_2 x_3 \bar{x}_4) \vee y_1 (\bar{x}_2 \bar{x}_3 \vee x_2 x_3 x_4) \vee (x_2 \bar{x}_3) \}
 \end{aligned} \tag{12}$$

[例終わり]

[0050] [定義10] (特性関数二分決定グラフ)

多出力論理関数 $F(X) = (f_0(X), f_1(X), \dots, f_{m-1}(X))$ の「特性関数二分決定グラフ (Binary Decision Diagram for Characteristic Function : BDD for CF)」とは、多出力論理関数 $F(X)$ の特性関数 $\chi(X, Y)$ を表現する二分決定グラフ (Binary Decision Diagram : BDD) をいう。但し、二分決定グラフの変数は、根節点を最上位としたとき、変数 y_j は f_j に依存する入力変数 $x_i \in \text{supp}(f_j)$ よりも下位に置く。ここに、 $\text{supp}(f_j)$ は f_j の依存変数の集合を表す。(非特許文献13, 14参照)

[定義終わり]

[0051] [例3]

図1(a)の真理値表によって表現される多出力論理関数の特性関数二分決定グラフは、図1(b)により表される。ここで、○は変数節点を表し、□は特性関数値の節点を表す。

[例終わり]

[0052] [定義11] (短絡除去)

特性関数二分決定グラフにおいて、出力を表す節点 $y_j (\in Y)$ から出る枝 (edge) のうち、定数0に接続する枝を取り除き、節点 y_j の親節点 (parent node) から節点 y_j の定数

0以外の子節点 (child node) へ直接枝を繋ぐ。この作業を、 y_j を表現する総ての節点に対して実行する操作を、「出力変数 y_j の短絡除去 (removal of the output variable y_j by shorting)」という。

[定義終わり]

[0053] 具体的に図2を用いて短絡除去の説明をする。図2(a)に示したような出力を表す節点 y_j ($\in Y$) を考える。節点 y_j の親節点を z_p 、節点 y_j の子節点を z_q とする。まず、特性関数値0に接続する節点を除去すると、図2(b)のようになる。次に、節点 y_j の親節点 z_p から子節点 z_q へ直接枝を繋ぐことにより、図2(c)のようにグラフが変形される。節点 y_j に関する短絡除去では、以上のような操作を、特性関数二分決定グラフの中にある出力を表す節点 y_j のすべてについて行うものである。

[0054] [定義12] (特性関数二分決定グラフの幅)

$(z_{n+m}, z_{n+m-1}, \dots, z_1)$ を特性関数二分決定グラフの変数順序とする。ここで、 z_{n+m} は、根節点に対応する。特性関数二分決定グラフの高さ k での「幅」とは、変数 z_k と変数 z_{k+1} との間の枝の本数をいう。ここで、同じ節点に接続している枝は一つと数え、定数0に向かう枝は無視する。また、特性関数二分決定グラフの高さ0での幅を1と定義する。

[定義終わり]

[0055] 変数 z_i の節点の高さ (level) とは、二分決定グラフにおいて順序づけられた変数において、終端節点から数えた順番をいう。但し、終端節点の高さは0とする。すなわち、 $n+m$ 個の変数 $\{z_i; i=1, \dots, n+m\}$ を有する二分決定グラフにおいて根節点から終端節点にかけて順序づけられた変数順序を $P=(z_{n+m}, z_{n+m-1}, \dots, z_1)$ とすると、変数 z_i の節点の高さは i となる。このとき、変数 z_i, z_j について $i > j$ のとき、変数 z_i は変数 z_j よりも高位 (high level) である (又は変数 z_j は変数 z_i よりも低位である) という。また、高さ i での幅は、変数 z_i と変数 z_{i+1} との間の枝の本数である。但し、同じ節点を特定するものは1つと数え、定数0に向かう枝は無視する。

[0056] [例4]

図1(b)の特性関数二分決定グラフにおいては、終端節点 (特性関数値がラベル付けられた節点) の高さが0、変数 y_2 の高さが1、変数 y_1 の高さが2、変数 x_4 の高さが3、

…、変数 x_1 の高さが7となる。また、高さ1での幅は、変数 y_1 の節点から変数 y_2 の節点へ向かう枝の本数(但し、終端節点0へ向かう枝は無視するものとする。)で2である。また、高さ2での幅は、変数 x_4 の節点から変数 y_1 の節点へ向かう枝の本数で4である。
[例終わり]

[0057] BDDを用いた関数分解において、BDDの幅は列複雑度 μ に等しい。([注意]列複雑度を考えるときは、元の多出力関数を考える(表3の分解表参照)。BDDを考えると、多出力関数の特性関数を考える。単一出力論理関数の時は、必然的に、出力変数は、一番下に来る。定数1に至るBDDの部分が、もとの関数のBDDと殆ど変わらない。)変数の分割(X_1, X_2)において、変数 X_1 の節点に直接接続されている変数 X_2 の節点は、分解表の列パターンに対応する。

[0058] 関数分解では、 $f(X_1, X_2)=g(h(X_1), X_2)$ と表現する。

[0059] [数13]

$$\lceil \log_2 \mu \rceil < |X_1| \quad (13)$$

が成立するとき、 X_1 を入力とし列関数の符号を生成する $h(X_1)$ の回路と、 $g(h(X_1), X_2)$ の回路を個別に実現することによって、関数 f を実現する(図3参照)。関数分解は分解後の関数 g の入力数が少ないほど効果的である。BDDの幅を削減できれば、列複雑度が減り、関数分解の効果が上がる。

[0060] [定義13] (ドント・ケアへの定数の割り当て)

2つの不完全定義列関数を $F^{(1)}=(f_0^{(1)}, f_1^{(1)}, \dots, f_{m-1}^{(1)})$, $F^{(2)}=(f_0^{(2)}, f_1^{(2)}, \dots, f_{m-1}^{(2)})$ とする。このとき、 $f_i^{(1)}=d$ 又は $f_i^{(2)}=d$ である $F^{(1)}$, $F^{(2)}$ の各要素に対して、次のような論理積演算 $f_i^{(1)} \cdot f_i^{(2)}$ を行うことを「ドント・ケアへの定数の割り当て(don't care assignment)」という。

[0061] [数14]

$$f_i^{(1)} \cdot f_i^{(2)} = \begin{cases} 1 & (\text{if } [f_i^{(1)} = 1 \wedge f_i^{(2)} = d] \vee [f_i^{(1)} = d \wedge f_i^{(2)} = 1]) \\ 0 & (\text{if } [f_i^{(1)} = 0 \wedge f_i^{(2)} = d] \vee [f_i^{(1)} = d \wedge f_i^{(2)} = 0]) \\ d & (\text{if } f_i^{(1)} = d \wedge f_i^{(2)} = d) \end{cases} \quad (14)$$

[定義終わり]

[0062] [定義14] (関数の両立)

二つの不完全定義関数 f_a と f_b にドント・ケアへの定数の割り当てを行うことで、同じ関数にできる場合、 f_a と f_b は「両立する (compatible)」といい、 $f_a \sim f_b$ と記す。

[定義終わり]

[0063] [補題1]

三つの不完全定義関数の特性関数を x_a 、 x_b とする。 $x_a \sim x_b$ 且つ $x_c = x_a x_b$ のとき、 $x_c \sim x_a$ 且つ $x_c \sim x_b$ が成立する。

[補題終わり]

[0064] [例5]

(表1)の分解表で、列関数の対 $\{\Phi_1, \Phi_2\}$, $\{\Phi_1, \Phi_3\}$, $\{\Phi_3, \Phi_4\}$ は両立する。列 Φ_1 と Φ_2 の論理積をとり、 Φ_1^* と Φ_2^* と置き換え、列 Φ_3 と Φ_4 の論理積をとり、 Φ_3^* と Φ_4^* と置き換えると(表3)のようになり、 $\mu = 2$ となる。

[0065] [表3]

		$X_1 = \{x_1, x_2\}$			
		00	01	10	11
$X_2 = \{x_3, x_4\}$	00	0	0	1	1
	01	1	1	d	d
	10	1	1	0	0
	11	0	0	0	0
		Φ_1^*	Φ_2^*	Φ_3^*	Φ_4^*

[例終わり]

[0066] [定義15] (両立グラフ)

関数を節点とし、両立する関数同士を枝で接続したグラフを「両立グラフ (compatible graph)」という。

[定義終わり]

[0067] [2] 本発明の基本的原理

(1) 不完全定義多出力論理関数の特性関数二分決定グラフの幅の削減

最初に、不完全定義多出力論理関数を表現する特性関数二分決定グラフの幅を削減する方法について説明する。一つの方法としては、特性関数二分決定グラフの1つの節点に着目し、その節点の2つの子節点が両立する場合、2つの子節点を併

合して一つにすることを繰り返し、節点数を削減する方法が考えられる。この方法のアルゴリズムは、以下のアルゴリズム1のようになる。

[0068] [アルゴリズム1]

特性関数二分決定グラフの根節点から、再帰的に以下の処理を行う。

- (1) 節点 v がドント・ケアを含まなければ終了;
- (2) 節点 v の2つの子節点 v_0, v_1 が両立するか調べる;
 - ・両立しなければ、 v_0, v_1 に対して再帰する;
 - ・両立すれば、 v_0, v_1 を v_{new} に置き換える。 v_{new} に対して再帰する。但し、 v_0, v_1, v_{new} を表す特性関数を x_0, x_1, x_{new} とすれば、 $x_{\text{new}} = x_0 \cdot x_1$ 。

[0069] [例6]

(表4)に4入力2出力不完全定義関数の真理値表を示す。

[0070] [表4]

x_1	x_2	x_3	x_4	f_0	f_1
0	0	0	0	d	1
0	0	0	1	d	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	d	d
0	1	0	1	d	d
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	d
1	1	0	1	1	d
1	1	1	0	d	0
1	1	1	1	d	1

[0071] (表4)の真理値表によって表される不完全定義多出力関数を特性関数二分決定グラフにより表すと、図4(a)のようになる。図4(a)では、節点1, 2が両立する子節点

を持っている。この特性関数二分決定グラフに対してアルゴリズム1を適用すると、図4(b)のようになる。この関数では、節点1を置き換える節点と、節点2を置き換える節点と同じになるので、図4(b)では、節点1, 2が節点3に置き換えられている。図4の特性関数二分決定グラフの左にあるWidthは、各高さでの幅である。最大幅は8から5に削減され、非終端節点の数は15から12に削減されている。

[例終わり]

[0072] アルゴリズム1のグラフ幅削減方法は、局所的な節点数の削減には効果がある。しかし、一つの節点の子節点同士でしか両立性を考えていないため、全体的な特性関数二分決定グラフの幅の削減にはあまり効果的ではない。そこで、次に、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を調べ、マッチングを行うことで直接幅を削減するグラフ幅削減方法を示す。

[0073] 関数分解において、すべての列関数の両立性を調べ、両立グラフを作成し、最小数の完全部分グラフ(クリーク)による節点被覆を行うことで、列複雑度 μ の最小化が可能である。しかしながら、この節点被覆の問題はNP困難であることが知られている(非特許文献16参照)。そこで、ここでは次のアルゴリズム2で表される発見的手法を使用する。

[0074] [アルゴリズム2] (クリーク集合による節点被覆)

与えられた両立グラフのすべての節点の集合を S_a とする。Cを節点集合の集合とする。枝を持たない節点を S_a から除去し、要素が1つの節点集合(クリーク)としてCの要素とする。

$S_a \neq \phi$ の間、以下の(1)〜(3)の操作を繰り返す:

(1) S_a の中で枝数最小の節点を調べ、これを v_i とする。 $S_i \leftarrow \{v_i\}$ とする。 S_a の中で v_i に接続している節点の集合を S_b とする;

(2) $S_b \neq \phi$ の間、以下の(a), (b)の操作を繰り返す;

(a) S_b の中で枝数最小の節点 v_j を調べ、 $S_i \leftarrow S_i \cup \{v_j\}$, $S_b \leftarrow S_b - \{v_j\}$ とする;

(b) v_j に接続していない節点を S_b から除去する;

(3) $C \leftarrow C \cup \{S_i\}$, $S_a \leftarrow S_a - S_i$.

[0075] 両立グラフ上におけるクリーク集合による節点被覆を利用して、次のアルゴリズム3

により、特性関数二分決定グラフの幅の削減を行うことができる。

[0076] [アルゴリズム3] (特性関数二分決定グラフの幅削減)

与えられた特性関数二分決定グラフの高さを t とし、定数節点の高さを0とする。高さ $t-1$ から1までの間、以下の操作を繰り返す：

- (1) 各高さにおけるすべての列関数の集合を作る；
- (2) 列関数の集合から両立グラフを作成する；
- (3) クリーク集合による節点被覆(例えば、アルゴリズム2)を行う；
- (4) 各クリークに被覆されているすべての節点に対応する列関数に対して、ドント・ケアへの定数の割り当てを行い、一つの列関数(割当列関数)にする；
- (5) 各列関数を(4)で作成した割当列関数に置換して、特性関数二分決定グラフを再構成する。

[0077] このグラフ幅削減方法によれば、特性関数のすべての部分関数の両立性を全体的に評価して最適なドント・ケアへの定数の割り当てができる。従って、特性関数二分決定グラフの幅を効果的に削減することが可能となる。

[0078] (2) 目的論理関数の関数分解とLUT回路の生成

次に、特性関数二分決定グラフからLUT論理回路を合成する手法について、その原理を説明する。本発明におけるLUT論理回路の合成は、特性関数二分決定グラフの分割と短絡除去とを利用して行う。特性関数二分決定グラフを分割して2つの回路を構成した場合、両回路を接続する接続線数は、次の[定理1]により算出することができる。

[0079] [定理1]

X_1, X_2 を入力変数の集合、 Y_1, Y_2 を出力変数の集合とする。特性関数二分決定グラフの変数順序を (X_1, Y_1, X_2, Y_2) とすると、特性関数二分決定グラフの高さ $|X_2| + |Y_2|$ における幅を W とする。ここで、 W を数える際、出力を表現する変数から、定数0へ向かう枝は無視する。多出力論理関数を図3の回路で実現する場合、二つの回路 H と G の間の必要かつ十分な接続線数 u は、式(15)により表される。

[0080] [数15]

$$u = \lceil \log_2 W \rceil \quad (15)$$

〔定理終わり〕

[0081] (証明)

特性関数二分決定グラフの構成法から、出力関数 Y_1 と Y_2 が、図3の回路で表現できることは明らかである。特性関数二分決定グラフにおいて、 Y_1 の出力を表現する節点を短絡除去すると、 Y_1 以外の関数を表現する特性関数二分決定グラフが得られる。また、この操作によって、特性関数二分決定グラフの幅が増えることはない。高さ $|X_2| + |Y_2|$ におけるBDDの幅を W とする。いま、関数分解 $g(h(X_1), X_2)$ の分解表を考えると、その列複雑度は W に等しい。従って、回路 H と回路 G の間の配線数は、

[0082] [数16]

$$\lceil \log_2 W \rceil \quad (16)$$

だけあれば十分である。また、分解表の列複雑度は W なので、二つのブロック間の配線は少なくとも式(16)の本数だけ必要である。

(証明終わり)

[0083] 特性関数二分決定グラフの分割及び短絡除去と上記〔定理1〕とを利用して、次のように目的論理関数の関数分解が行われる。

[0084] まず、目的論理関数 $F=(f_0, f_1, \dots, f_{m-1})$ の特性関数二分決定グラフの変数順序が (X_1, Y_1, X_2, Y_2) 、 $Y_1=(y_0, y_1, \dots, y_{k-1})$ 、 $Y_2=(y_k, y_{k+1}, \dots, y_{m-1})$ であるとする。この場合、 $y_j=f_j(X_1)$ ($j=0, 1, \dots, k-1$)は図3の回路 H で直接実現する。一方、高さ $|X_2| + |Y_2|$ における特性関数二分決定グラフの幅を W とする。 W 本の枝に u ビット(u は式(15)により得られる数。)の異なる二進数を割り当てる。二つのブロック H, G 間を接続する枝が実現する関数を h_1, h_2, \dots, h_u とすると、ブロック G の出力関数 Y_2 は $(h_1, h_2, \dots, h_u, X_2)$ を入力変数とする論理関数として表現可能であり、その特性関数二分決定グラフは図5のようになる。

[0085] [例7]

本例では、2つの2ビットの2進数を加算する回路(ADR2)を、中間出力を有する関数分解を用いて実現する場合について説明する。ADR2の入出力の関係は式(17)のように定義できる。これより、 s_0, s_1, s_2 は、それぞれ式(18)により表される。また、真理値表は図6(a)により表される。

[0086] [数17]

$$\begin{array}{r}
 a_0 \\
 b_0 \\
 \hline
 s_2 s_0
 \end{array} \quad (17)$$

[0087] [数18]

$$\begin{aligned}
 s_0 &= a_0 \oplus b_0 \\
 s_1 &= a_0 b_0 \oplus (a_1 \oplus b_1) \\
 s_2 &= a_0 b_0 (a_1 \vee b_1) \vee a_1 b_1
 \end{aligned} \quad (18)$$

[0088] 変数の分割として、 $X_1=(a_0, b_0)$, $Y_1=(s_0)$, $X_2=(a_1, b_1)$, $Y_2=(s_1, s_2)$ とおく。このとき、変数順序は、 $(X_1, Y_1, X_2, Y_2)=(a_0, b_0, s_0, a_1, b_1, s_1, s_2)$ となる。従って、ADR2の特性関数二分決定グラフは、図6(b)により表される。 $Z=(Z_A, Z_B)$, $Z_A=(X_1, Y_1)$, $Z_B=(X_2, Y_2)$ のように分割し、 Z_A を図3の回路H、 Z_B を図3の回路Gにより構成する。このとき、高さ $|Z_B|$ での特性関数二分決定グラフの幅Wは2となる。従って、回路Hと回路Gとを連結する線は、式(15)より、1本あればよい。出力 s_0 は、 X_1 のみの関数として表現することが可能である。

[0089] また、変数組 Z_A により構成される特性関数二分決定グラフは、図7(a)のようになる。これは、図6の特性関数二分決定グラフの分割線よりも根節点側(上側)の部分を取り取ったものである。尚、図7(a)では特性関数値0につながる枝は省略してある。回路Hと回路Gとを連結する線は1本なので、図7(a)のグラフの終端節点には、1ビットの中間変数 h_1 を導入して、各終端節点に割り当てる。なお、変数 h_1 への符号の割り当ては任意である。この特性関数二分決定グラフは、容易に多端子二分決定グラフ(MTBDD)に変換することができる。図7(a)のグラフを変換することにより得られるMTBDDは、図7(b)のようになる。図7(b)のMTBDDから回路Hに相当するLUTを図7(c)のように生成することができる。

[0090] 次に、図6の特性関数二分決定グラフについて、先に導入した新しい中間変数 h_1 を用いて、図6(b)に示した特性関数二分決定グラフの分割線よりも上部を、 h_1 を入力変数とする決定木(decision tree)で置き換える。これにより、特性関数二分決定グラフは図8(a)のように変形される。尚、図8(a)でも特性関数値0につながる枝は必要ないので省略してある。この特性関数二分決定グラフは、容易にMTBDDに変換す

ることができる。図8(a)のグラフを変換することにより得られるMTBDDは、図8(b)のようになる。更に、図8(b)のMTBDDから回路Gに相当するLUTを図9(a)のように生成することができる。従って、結局、図9(b)に示したようなLUTカスケードを構成することができる。

〔例終わり〕

[0091] [3] 本発明の構成及び作用

本発明に係るグラフ幅削減装置の第1の構成は、入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(19)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフの幅を削減するグラフ幅削減装置であって、以下の構成を備えていることを特徴とする:

多出力論理関数 $F(X)$ の特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルを記憶する節点テーブル記憶手段;

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定手段;

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定手段により設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成手段;

及び、前記列関数生成手段が生成する各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数(以下「割当列関数(assigned column function)」という。)とするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テ

ブルを更新する割当BDD再構成手段。

[0092] [数19]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{ \bar{y}_i f_{i,0} \vee y_i f_{i,1} \vee f_{i,d} \} \quad (19)$$

但し、 $f_{i,0}$, $f_{i,1}$, $f_{i,d}$ は、それぞれ、式(20)で定義されるOFF関数、ON関数、及びDC関数を表す。

[0093] [数20]

$$f_{i,0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (20)$$

[0094] この構成により、分割線設定手段が分割の高さlevを設定すると、列関数生成手段が特性関数二分決定グラフを分割の高さlevで分割し関数分解することによって得られる列関数を生成する。そして、割当BDD再構成手段は、各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の割当列関数とする。それとともに、これらの新たな割当列関数を用いて特性関数二分決定グラフ再構成し節点テーブル記憶手段の節点テーブルを更新する。これにより、分割の高さlevにおける特性関数二分決定グラフの幅が削減される。

[0095] 一般に、二分決定グラフを用いて関数分解を行う場合、二分決定グラフの幅が小さいほど、関数分解後の論理回路も小さくなる。従って、上述のようなドント・ケアへの定数の割り当てを実行することによって、二分決定グラフの幅を削減し、関数分解後の論理回路を小さくすることができる。

[0096] 本発明に係るグラフ幅削減装置の第2の構成は、前記第1の構成において、前記列関数を節点(以下「関数節点」という。)とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝(以下「両立枝」という。)により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ(以下「関数節点データ」という。)のテーブルとして記憶するための両立グラフ記憶手段;

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数が

ら、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成手段；

及び、前記両立グラフのすべての節点について、完全部分グラフ（以下「クリーク」という。）による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成手段；

を備え、

前記列関数生成手段は、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定手段により設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するものであり、

前記割当BDD再構成手段は、前記クリーク生成手段が生成する前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新するものであることを特徴とする。

[0097] この構成によれば、列関数生成手段が、節点データ記憶手段に記憶された節点テーブルから、分割の高さlevの節点の各枝に対応する列関数を生成する。そして、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存する。両立枝生成手段は、両立グラフ記憶手段に記憶された各関数節点データに対する列関数から、両立する列関数の組を選出する。そして、選出された両立する列関数に対する関数節点データに、それらの関数節点同士を連結する両立枝を追加して関数節点データの更新を行う。次に、クリーク生成手段は、両立グラフのすべての節点について、最小数のクリークによる節点被覆を行い、クリーク・データを生成する。最後に、割当BDD再構成手段は、各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とする。そして、割当列関数により、特性関数二分決定グラフを再構成し、節点テーブルを更新する。

- [0098] 以上のようにして、上述のアルゴリズム3の(1)ー(5)を実行し、列関数の両立グラフのクリーク被覆による特性関数二分決定グラフの幅の削減を行うことができる。本発明によれば、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を、両立グラフを用いて全体的に評価して、ドント・ケアへの定数の割り当てを行うことができるため、極めて効果的に特性関数二分決定グラフの幅を削減することが可能となる。
- [0099] ここで、クリーク生成手段は、クリークを生成する場合に、両立グラフのすべての節点について、最小数のクリークにより節点被覆を行うことが望ましい。しかしながら、前述したように、最小数のクリークによる節点被覆問題はNP困難であり実用的ではない。従って、実際にはクリーク生成手段は、発見的方法によってできるだけ少ないクリーク数で節点被覆を行うことができるようにクリークを生成するように構成するのがよい。
- [0100] 本発明に係るグラフ幅削減装置の第3の構成は、前記第1又は第2の構成において、前記分割線設定手段は、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから下位に向かって順次分割の高さlevの設定を行うものであり、前記割当BDD再構成手段は、前記分割線設定手段が設定する前記各分割の高さlevにおいて逐次前記特性関数二分決定グラフ再構成を行うことを特徴とする。
- [0101] この構成によれば、特性関数二分決定グラフの根節点の高さtの子節点の高さt-1から定数節点の上の節点の高さ1まで、上述したアルゴリズム3による特性関数二分決定グラフの幅の削減を行うことができる。従って、多出力論理関数のすべてのサポート変数に対して効果的に特性関数二分決定グラフの幅を削減することが可能となる。
- [0102] 本発明に係る論理回路合成装置の第1の構成は、入力変数を $X=(x_1, \dots, x_n)$ ($n \in$ 自然数)とする多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ の特性関数二分決定グラフから、前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを生成する論理回路合成装置であって、以下の構成を備えていることを特徴とする：

完全定義関数である前記多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(21)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段;

前記ルックアップ・テーブルを記憶するLUT記憶手段;

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定手段;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($\in Y$)に関わる節点 v_j 及びその親節点 v_k の節点データについて、当該節点 v_j の枝 $e_0(v_j)$, $e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k)$, $e_1(v_k)$ のうち当該節点 v_j を特定する枝 $e_c(v_k)$ を、当該節点 v_j の前記枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う短絡除去手段;

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し(但し、同じ節点を特定するものは1つと数え、定数0に向かう枝は無視する。)、その個数を前記分割の高さlevの分割線における幅Wとして出力するBDD幅計測手段;

前記BDD幅計測手段が出力する幅Wに基づき、式(22)の演算により中間変数の個数uを算出する中間変数算出手段;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1

に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納するLUT生成手段;

及び、前記中間変数算出手段が算出する前記中間変数の個数 u と等しい制御入力数を有する二分木(binary tree)を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するBDD再構成手段。

[0103] [数21]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X)) \quad (21)$$

[0104] [数22]

$$u = \lceil \log_2 W \rceil \quad (22)$$

[0105] この構成によれば、

(a)まず、論理回路合成を行う目的論理関数の特性関数二分決定グラフの節点データを節点テーブル記憶手段に格納しておく。短絡除去手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さ lev の分割線に対して根節点を含む側の部分グラフ B_0 について短絡除去処理を行う。短絡除去処理については、上記「[2]本発明の基本的原理」の欄において説明した通りである。BDD幅計測手段は、上記短絡除去処理が行われた結果得られる特性関数二分決定グラフについて、上記分割線における幅 W を計測する。そして、中間変数算出手段は、式(22)によって中間変数の個数 u を算出する。

[0106] (b)次に、LUT生成手段は、節点テーブル記憶手段に格納された節点データで構成される特性関数二分決定グラフについて、所定の高さ lev の分割線に対して根節点を含む側の部分グラフ B_0 について、ルックアップ・テーブルを生成する。この生成方法の原理については、上記「[1]本発明の基本的原理」の欄において説明した通り

である。このLUTの出力は、部分グラフ B_0 に属する出力変数 $Y_0=(y_0, \dots, y_{k-1})$ 及び u 個の中間変数 $H=(h_1, \dots, h_u)$ となる。生成されたルックアップ・テーブルは、LUT記憶手段に格納される。中間変数 H の節点に対しては、適当な符号を割り当てることができる。

- [0107] (c)最後に、BDD再構成手段は、 u 個の制御入力数を有する二分木を生成し、各制御入力数に対して上記中間変数 $H=(h_1, \dots, h_u)$ を対応させる。そして、BDD再構成手段は、この処理により再構成された特性関数二分決定グラフの各節点の節点データを節点テーブル記憶手段に格納し、節点テーブル記憶手段が保持する特性関数二分決定グラフを更新する。
- [0108] 以上のような(a)ー(c)の処理を繰り返せば、目的論理関数は複数の部分関数に関数分解され、LUTカスケード論理回路が構成される。また、LUT生成手段により生成されるルックアップ・テーブルを特性関数二分決定グラフにして、更に同様の処理によって関数分解を行うことによって、LUTネットワーク論理回路が構成される。
- [0109] 本構成による論理回路合成装置では、分割線の位置を、部分グラフ B_0 の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフ B_0 に関する変数を束縛変数とした分解表の列複雑度が極度に増大することを防止することが可能である。従って、比較的少容量のメモリを用いて論理回路合成装置を実現することが可能である。また、部分グラフ B_0 に関する変数を束縛変数とした分解表の列複雑度を抑えることができるため、計算処理量やメモリ・アクセス回数を少なくすることが可能であり、論理関数分解処理の演算処理速度を高速化することができる。
- [0110] 尚、短絡除去手段による短絡除去処理において、実際に節点テーブルの構造自体を書き換えることによって短絡除去処理を行ってもよいが、節点テーブルのデータ構造は変化させず、特性関数に二分決定グラフを解釈するプログラムにおいて或る節点に短絡処理がなされたものとみなすようにすることもできる。例えば、短絡処理がなされた出力変数の節点については節点ラベルを特定の値に書き換える。そうすることにより、節点ラベルを調べることで、その変数が出力か否かが判定できるようにして、事実上短絡処理がなされたと見なすことができるようにすることができる。
- [0111] 本発明に係る論理回路合成装置の第2の構成は、前記第1の構成において、前記節点テーブル記憶手段には、出力にドント・ケアを含む不完全定義関数である前記

多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(19)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in$ 自然数)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶されていることを特徴とする。

[0112] この構成により、不完全定義多出力関数についても、完全定義多出力関数と同様、LUTカスケード論理回路やLUTネットワーク論理回路を構成することが可能となる。

[0113] 本発明に係る論理回路合成装置の第2の構成は、前記第2の構成において、前記第1乃至3の何れか一の構成のグラフ幅削減装置を備え、前記短絡除去手段は、前記グラフ幅削減装置によって前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を削減し更新された前記節点テーブルについて短絡除去処理を行うものであることを特徴とする。

[0114] この構成により、不完全定義関数の特性関数二分決定グラフの幅を削減した上でLUTカスケードの合成を行うため、不完全定義多出力関数の冗長性を削減して、LUTカスケードのメモリ使用効率を改善することができる。

[0115] 本発明に係る論理回路合成装置の第4の構成は、前記第1乃至3の何れか一の構成において、

前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。)とし、論理関数 $f_j(\in F(X))$ が依存する入力変数の集合を $\text{supp}(f_j)$ としたとき、式(23)で表される T の値が最小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定手段;

前記各入力変数 $x_i(\in X)$ 及び出力を表す変数 $y_j(\in Y)$ の順序を、式(24)を満たす順序 P に決定する全変数順序決定手段;

及び、前記全変数順序決定手段で決定された順序 P に従って、特性関数二分決定

グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成手段
 ;
 を備えていることを特徴とする。

[0116] [数23]

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{supp}(f_{\pi[l]}) \right| \quad (23)$$

[0117] [数24]

$$P = \left(\text{supp}(f_{\pi[0]}), y_{\pi[0]}, \text{supp}(f_{\pi[1]}) - \text{supp}(f_{\pi[0]}), y_{\pi[1]}, \text{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^1 \text{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \cdots, \text{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \text{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \quad (24)$$

[0118] 実用的な多出力論理関数の特性関数二分決定グラフは、通常はかなりの大きさとなる。そのため、出力を分割する方法が必要である。そこで、上記構成によれば、出力変数順序決定手段が、論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を式(23)で表される T の値が最小となるように決定することにより、BDD生成手段が生成する特性関数二分決定グラフの節点データの数を削減できる。特性関数二分決定グラフにおいて、各出力変数 $y_j = f_j(X)$ に対して、その出力変数が依存する入力変数が、その出力変数 y_j よりも上位に置かれるため、依存変数になるべく増えないような順序に並べられ、一つずつ出力を増やしながら特性関数二分決定グラフが作られる。このようにして、出力を最適に分割する。このとき、特性関数二分決定グラフの節点データの数が削減できるので、その後の論理回路合成に要する処理時間が短縮され、高速な論理回路合成が可能となる。

[0119] 本発明に係るグラフ幅削減方法の第1の構成は、入力変数を $X = (x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X) = (f_0(X), \dots, f_{m-1}(X))$ に対して式(19)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y = (y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端

節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段を備えたシステムにおいて、当該特性関数二分決定グラフの幅を削減するグラフ幅削減方法であって、以下の各ステップを有することを特徴とする：

前記節点テーブルによって表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定ステップ；

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定ステップで設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成ステップ；

及び、前記列関数生成ステップにおいて生成される各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数(以下「割当列関数」という。)にするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当BDD再構成ステップ。

[0120] 本発明に係るグラフ幅削減方法の第2の構成は、前記第1の構成において、前記システムは、前記列関数を節点(以下「関数節点」という。)とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝(以下「両立枝」という。)により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ(以下「関数節点データ」という。)のテーブルとして記憶するための両立グラフ記憶手段を備えており、

前記列関数生成ステップにおいては、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定ステップで設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するとともに、

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データ

にそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成ステップ;

及び、前記両立グラフのすべての節点について、完全部分グラフ(以下「クリーク」という。)による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成ステップ;

を有し、

前記割当BDD再構成ステップにおいては、前記クリーク生成ステップにおいて生成される前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新することを特徴とする。

[0121] 本発明に係るグラフ幅削減方法の第3の構成は、前記第1又は第2の構成において、前記分割線設定ステップにおける前記分割の高さlevを、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから最下位の高さまで順次変更しながら、前記分割線設定ステップ乃至前記割当BDD再構成ステップを実行することを特徴とする。

[0122] 本発明に係る論理回路合成方法の第1の構成は、入力変数を $X=(x_1, \dots, x_n)$ ($n \in$ 自然数)とする完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(21)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in$ 自然数)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段;

及び、ルックアップ・テーブルを記憶するためのLUT記憶手段;

を備えたシステムにおいて、前記特性関数二分決定グラフから、前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを合成する論理回路合成方法であって、以下のステップを有することを特徴とする:

前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定ステップ;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 $y_r (\in Y)$ に関わる節点 v_j 及びその親節点 v_k の節点データについて、当該節点 v_j の枝 $e_0(v_j), e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k), e_1(v_k)$ のうち当該節点 v_j を特定する枝 $e_c(v_k)$ を、当該節点 v_j の前記枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を実行する短絡除去ステップ;

前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し(但し、同じ節点を特定するものは1つと数え、定数0に向かう枝は無視する。)、その個数を前記分割の高さlevの分割線における幅Wとして出力するBDD幅計測ステップ;

前記幅Wに基づき、式(22)の演算により中間変数の個数uを算出する中間変数算出ステップ;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納するLUT生成ステップ;

及び、前記中間変数算出ステップにおいて算出された前記中間変数の個数uと等しい制御入力数を有する二分木を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するBDD再構成ステップ。

- [0123] 本発明に係る論理回路合成方法の第2の構成は、前記第1の構成において、前記節点テーブル記憶手段には、入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(19)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i), e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶されていることを特徴とする。
- [0124] 本発明に係る論理回路合成方法の第3の構成は、前記第2の構成において、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を、前記第1乃至3の何れかの構成のグラフ幅削減方法によって削減し、前記節点テーブル記憶手段に記憶された前記節点テーブルを更新した後に、前記分割線設定ステップ乃至前記BDD再構成ステップを実行することを特徴とする。
- [0125] 本発明に係る論理回路合成方法の第4の構成は、前記第1乃至3の何れかの構成において、前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f_j が i 番目であることを表す。)とし、論理関数 $f_j (\in F(X))$ が依存する入力変数の集合を $\text{supp}(f_j)$ としたとき、式(23)で表される T の値が最小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定ステップ；
- 前記各入力変数 $x_i (\in X)$ 及び出力を表す変数 $y_j (\in Y)$ の順序を、式(24)を満たす順序 P に決定する全変数順序決定ステップ；
- 及び、前記順序 P に従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成ステップ；
- を実行した後に、前記分割線設定ステップ乃至前記BDD再構成ステップを実行することを特徴とする。
- [0126] 本発明に係るプログラムの第1の構成は、コンピュータに第1乃至3の何れかの構成

のグラフ幅削減方法を実行させることを特徴とする。

[0127] 本発明に係るプログラムの第2の構成は、コンピュータに第1乃至4の何れかの構成の論理回路合成方法を実行させることを特徴とする。

[0128] 本発明に係るプログラマブル論理回路の構成は、第1乃至4の何れかの構成の論理回路合成方法により論理合成されたことを特徴とする。

発明の効果

[0129] 以上のように、本発明に係るグラフ幅削減装置及びグラフ幅削減方法によれば、特性関数二分決定グラフの幅を構成するすべての部分関数の両立性を調べてマッチングを行うことができるため、効果的に特性関数二分決定グラフの幅を削減することが可能となる。

また、本発明に係る論理回路合成装置及び論理回路合成方法によれば、特性関数二分決定グラフを分割する分割線の位置を、部分グラフ B_0 の大きさが過度に大きくならないように適度な高さに決めることで、部分グラフ B_0 に関する変数を束縛変数とする分解表の列複雑度が極度に増大するのを防止することが可能となる。その結果、比較的小容量のメモリを用いて論理回路合成を実現でき、また、論理関数分解処理の演算処理速度を高速化することができる。従って、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数の分解によるLUT論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。また、本発明は、任意の多出力論理関数のLUT論理回路生成に対して汎用的に適用可能である。

図面の簡単な説明

[0130] [図1]特性関数二分決定グラフの一例を示す図である。

[図2]出力変数の短絡除去を説明するための図である。

[図3]多出力論理関数を2つの関数に関数分解した場合の論理回路の構成を表す図である。

[図4]表4の不完全定義関数の特性関数二分決定グラフに対してアルゴリズム1による幅の削減を行う例を表す図である。

[図5]短絡除去を行った後の特性関数二分決定グラフの構成を表す図である。

[図6]ADR2の特性関数二分決定グラフを表す図である。

[図7]ADR2の特性関数二分決定グラフを分割したときの根節点側の部分グラフをLUTに変換する処理を説明するための図である。

[図8]ADR2の特性関数二分決定グラフを短絡除去し、LUTに変換する処理を説明するための図である。

[図9]ADR2の特性関数二分決定グラフを短絡除去し、LUTに変換する処理を説明するための図である。

[図10]本発明の実施例1に係る論理回路合成装置及びその周辺装置の構成を表す図である。

[図11]実施例1における論理回路合成方法の全体の流れを示すフローチャートである。

[図12]変数順序の決定処理の流れを表すフローチャートである。

[図13]節点テーブルのデータ構造を表す図である。

[図14]式(31)の既約な特性関数二分決定グラフ及びその節点テーブルを表す図である。

[図15]式(31)の特性関数二分決定グラフ及びその節点テーブルを表す図である。

[図16]LUTカスケード論理回路の合成処理の流れを表すフローチャートである。

[図17]本発明の論理回路合成方法によるLUTカスケード論理回路の合成についての実験結果を示す図である。

[図18]本発明の実施例2に係る論理回路合成装置及びその周辺装置の構成を表す図である。

[図19]図18のグラフ幅削減装置20の構成を表す図である。

[図20]グラフ幅削減装置20のグラフ幅削減処理の全体の流れを表すフローチャートである。

[図21]クリーク集合による両立グラフの節点被覆処理の流れを表すフローチャートである。

[図22]例10の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。

[図23]例10の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。

[図24]例10の不完全定義関数の特性関数二分決定グラフの幅削減処理を説明する図である。

[図25]ベンチマーク関数5-7-11-13 RNSについてLUTカスケード論理回路の合成を行った結果を比較する図である。

符号の説明

- [0131] 1, 1' 論理回路合成装置
- 2 論理仕様記憶手段
 - 3 入力装置
 - 4 出力装置
 - 5 出力変数順序決定手段
 - 6 全変数順序決定手段
 - 7 BDD生成手段
 - 8 節点テーブル記憶手段
 - 9 変数順序最適化手段
 - 10 分割線設定手段
 - 11 短絡除去手段
 - 12 BDD幅計測手段
 - 13 中間変数算出手段
 - 14 LUT生成手段
 - 15 BDD再構成手段
 - 16 LUT記憶手段
 - 20 グラフ幅削減装置
 - 21 BDD高さ計測手段
 - 22 分割線設定手段
 - 23 列関数生成手段
 - 24 列関数記憶手段

- 25 両立枝生成手段
- 26 両立グラフ記憶手段
- 27 クリーク生成手段
- 28 クリーク・データ記憶手段
- 29 割当BDD再構成手段

発明を実施するための最良の形態

[0132] 以下、本発明を実施するための最良の形態について、図面を参照しながら説明する。

実施例 1

[0133] 図10は本発明の実施例1に係る論理回路合成装置及びその周辺装置の構成を表す図である。実施例1に係る論理回路合成装置1は、論理仕様記憶手段2に格納された論理回路の論理仕様から、LUT論理回路を合成し、出力装置4に出力する。論理仕様記憶手段2には、入力装置3により、目的論理関数がネットリストなど論理仕様データとして格納される。

[0134] 尚、本実施例においては、論理回路設計用のCAD装置において論理回路合成装置1を使用する場合を想定している。論理回路合成装置1は、プログラム形態として提供され、そのプログラムをコンピュータに読み込んで実行することによって、図10に示したような論理回路合成装置1の機能構成がコンピュータで実現される。尚、図10において、入力装置3は、キーボード、マウス、CD-ROMドライブ等の各種データ入力装置を表している。出力装置4は、ディスプレイ、プリンタ、磁気記録装置等の各種データ出力装置を表している。また、出力装置4として、プログラマブル論理回路に生成されたLUT論理回路を直接書き込むためのプログラム書込装置を使用することもできる。

[0135] 論理回路合成装置1は、出力変数順序決定手段5、全変数順序決定手段6、BDD生成手段7、節点テーブル記憶手段8、変数順序最適化手段9、分割線設定手段10、短絡除去手段11、BDD幅計測手段12、中間変数算出手段13、LUT生成手段14、BDD再構成手段15、及びLUT記憶手段16を備えた構成からなる。

[0136] 出力変数順序決定手段5は、多出力論理関数 $f(X)$ の要素の順序 $\pi = (\pi[0], \dots,$

$\pi[m-1])$ を決定する。全変数順序決定手段6は、各入力変数 $x_i (\in X)$ 及び出力を表す変数 $y_j (\in Y)$ の順序を、所定の順序Pに決定する。BDD生成手段7は、全変数順序決定手段6で決定された順序Pに従って、特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段8に格納される。変数順序最適化手段9は、節点テーブル記憶手段8に格納された節点テーブルの変数順序を更に最適化する。

[0137] 分割線設定手段10は、節点テーブル記憶手段8に格納された特性関数二分決定グラフの節点データに対して、当該特性関数二分決定グラフを分割する分割線の高さlevの決定を行う。短絡除去手段11は、節点テーブル記憶手段8に格納された特性関数二分決定グラフに対し、高さlevの分割線より上の部分グラフの短絡除去処理を行う。BDD幅計測手段12は、短絡除去処理がされた特性関数二分決定グラフの分割線における幅Wとして出力する。中間変数算出手段13は、BDD幅計測手段12が出力する幅Wに基づき中間変数の個数uを算出する。LUT生成手段14は、分割線において2つの部分グラフ B_0, B_1 に分割したときの根節点を含む側の部分グラフ B_0 に属するものについて、LUTを生成しそれをLUT記憶手段16に格納する。BDD再構成手段15は、部分グラフ B_0 を中間変数の個数uと等しい制御入力数を有する二分木に置き換えて、特性関数二分決定グラフを再構成し、節点テーブル記憶手段8を更新する。

[0138] 以上のように構成された本実施例に係る論理回路合成装置1について、以下その動作を説明する。

[0139] 図11は実施例1における論理回路合成方法の全体の流れを示すフローチャートである。全体的な処理の流れでは、最初に、出力変数順序決定手段5と全変数順序決定手段6が、論理仕様記憶手段2に格納された論理仕様を読み出す。そして、読み出された論理仕様から、入力変数Xと出力変数Yとを抽出し、これら各変数の初期順序の決定を行う(S1)。次に、BDD生成手段7は、論理仕様記憶手段2から論理仕様データを読み出し、上記決定された変数の初期順序に従って特性関数二分決定グラフの節点テーブルを生成する。生成された節点テーブルは、節点テーブル記憶手段8に格納される(S2)。次に、変数順序最適化手段9は、節点テーブル記憶手段8

に格納された節点テーブルにおいて、特性関数二分決定グラフの幅の総和がより小さくなるように、更に変数順序の入れ替えを行い、変数順序を最適化する(S3)。このとき、変数順序の入れ替えは、出力変数の順序集合 $Y=(y_0, y_1, \dots, y_{m-1})$ における出力変数 y_j ($j=0, 1, \dots, m-1$)に対して、その出力変数 y_j が依存する入力変数 x_i ($i \in \text{supp}(f_j)$)は、出力変数 y_j よりも常に高位であるという条件のもとで、変数の入れ替えが行われる。最後に、節点テーブル記憶手段8に格納された特性関数二分決定グラフの節点テーブルに基づいて、LUT論理回路の合成が行われる(S4)。以下、上記各段階での処理について詳述する。

[0140] (1)変数X,Yの初期順序の決定

図12は変数順序の決定処理の流れを表すフローチャートである。まず、出力変数順序決定手段5は、内部変数 i, j, renew を0に初期化する(S10)。 i, j は置換する変数のインデックスを表す内部変数であり、 renew は順序の置換が行われたことを検出するための更新フラグである。また、出力変数順序決定手段5は、出力変数Yの変数順序を表す m 個の要素を持つ順序 $\pi_0=(\pi_0[0], \pi_0[1], \dots, \pi_0[m-1])$ を $(0, 1, \dots, m-1)$ に初期化する(S2)。順序 π_0 は、出力変数Yの変数順序を表す配列である。出力変数の順序は、順序 π_0 により、式(25)のように順序づけされる。

[0141] [数25]

$$Y = (y_{\pi_0[0]}, y_{\pi_0[1]}, \dots, y_{\pi_0[m-1]}) \quad (25)$$

[0142] 次に、出力変数順序決定手段5は、式(26)の演算を行うことにより、順序 π_0 に対する出力変数順序評価関数を算出し、これを変数 T_{\min} に格納する(S12)。ここで、出力変数順序評価関数とは、式(26)の右辺の関数である。この出力変数順序評価関数は、各出力変数Yの順序 π_0 が、依存変数の数の少ない順に出力変数Yを順序づけるものであるときに最小となる。従って、出力変数順序評価関数を最小化することにより、出力変数Yの順序は最適化されることとなる。また、 $\text{supp}(f_j)$ は、論理関数 $f_j(\in f(X))$ が依存する入力変数の集合を表す。例えば、 $f_j=f_j(x_1, x_2, x_5, x_{10})$ であれば、 $|\text{supp}(f_j)|=4$ である。また、 $|\text{supp}(f_j)|$ は $\text{supp}(f_j)$ の変数の個数である([定義4]参照)。

[0143] [数26]

$$T_{min} = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{supp}(f_{\pi_0[l]}) \right| \quad (26)$$

[0144] 次に、出力変数順序決定手段5は、 $i \neq j$ であるかどうかを判定する(S13)。 $i=j$ の場合には、ステップS19に行く。これは、 $i=j$ の場合には置換が行われないので、 $i=j$ の場合を除外する必要があるからである。

[0145] ステップS13において、 $i \neq j$ であれば、まず、変数置換後の順序 π_1 を初期順序 π_0 に初期化した後(S14)、順序 π_1 の要素 $\pi_1[i]$ と要素 $\pi_1[j]$ との置換を行う(S15)。そして、出力変数順序決定手段5は、式(27)の演算を行うことにより、順序 π_1 に対する出力変数順序評価関数Tを算出する(S16)。

[0146] [数27]

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{supp}(f_{\pi_1[l]}) \right| \quad (27)$$

[0147] ここで、置換後の出力変数順序評価関数Tが置換前の出力変数順序評価関数 T_{min} よりも小さい($T < T_{min}$)場合には(S17)、順序 π_1 のほうが出力変数の順序としてはより適当であると判断されるため、出力変数順序決定手段5は、順序 π_0 を π_1 に更新し、 T_{min} をTとする。そして、順序 π_0 の更新がされたことを表す更新フラグrenewを1とする(S18)。

[0148] 次に、 $j < m-1$ であれば(S19)、jを1だけインクリメントして(S20)、ステップS13に戻り、上記出力変数の順序の最適化処理を繰り返す。

[0149] ステップS18で $j=m-1$ ならば、i番目の出力変数の順序を固定したときの順序の入れ替えに対する評価がすべて終了したことになるので、jを0に初期化する(S21)。

[0150] 次に、 $i < m-1$ であれば(S22)、次の出力変数の順序を固定したときの順序の入れ替えに対する評価を行うべく、iを1だけインクリメントして(S23)、ステップS13に戻る。

[0151] ステップS22において、 $i=m-1$ の場合、i番目の出力変数の順序を固定した場合についての評価は一通り終了したことになる。但し、順序入れ替えの総数は、 $m!$ 通りであるが、この段階では、順序入れ替えの評価は $m \times (m-1)$ 通りしか行われていない。そこで、残りの場合についても評価するかどうかを判断するため、上記 $m \times (m-1)$ 通り

の評価において順序 π_0 の更新が行われたかどうかを判定する。すなわち、変数 renew が 1 であるかどうかを参照する (S24)。ここで、 $\text{renew}=1$ であれば、 i を 0、 renew を 0 に初期化して (S25)、再びステップ S13 の処理に戻る。尚、ステップ S24 で $\text{renew}=0$ であれば、終了する。

[0152] 以上のようにして、出力変数 Y の順序 π_0 は、出力変数順序評価関数 T_{\min} の値が極小化する順序に最適化される。出力変数順序決定手段 5 は、出力変数 Y の順序を順序 π_0 に並べ替えて出力する。

[0153] 尚、上記処理では、順序入れ替えの評価は必ずしも $m!$ 回行われないので、必ずしも出力変数順序評価関数 T_{\min} の値が最小となるように最適化されるとは限らないことに注意しておく。 $m!$ 回のすべてについての評価を行わないこととしたのは、処理時間を短縮するためである。しかしながら、上記変数並べ替えのアルゴリズムは一例であり、計算処理速度が十分速い場合や、出力変数及び入力変数の数が比較的少ない場合には、 $m!$ 回のすべてについて総当たりで評価を行うことにより、出力変数順序評価関数 T_{\min} の値が最小となるように最適化してもよい。

[0154] 出力変数の順序が決定された後、全変数順序決定手段 6 は、入力変数 X 及び出力変数 Y の順序を式 (28) の順序 P に従って決定する。

[0155] [数28]

$$P = \left(\text{supp}(f_{\pi[0]}), y_{\pi[0]}, \text{supp}(f_{\pi[1]}) - \text{supp}(f_{\pi[0]}), y_{\pi[1]}, \text{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^1 \text{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \dots, \text{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \text{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \quad (28)$$

ここで、式 (28) の条件を満たす限りにおいては、入力変数 X の順序の決め方は任意である。したがって、ここでは、出力変数

[0156] [数29]

$$y_{\pi[j]} \quad (j = 0, 1, \dots, m-1) \quad (29)$$

の前にある入力変数の集合

[0157] [数30]

$$(x_{\theta[p]}, x_{\theta[p+1]}, \dots, x_{\theta[p+q-1]}) = \text{supp}(f_{\pi[j]}) - \left(\sum_{k=0}^{j-1} \text{supp}(f_{\pi[k]}) \right) \quad (30)$$

の要素間の順序については、 x_i のインデックス i が小さい順に順序づけることとする。
尚、式(30)において $(\theta[1], \theta[2], \dots, \theta[p], \theta[p+1], \dots, \theta[p+q-1], \dots, \theta[n])$ は、入力変数 X の順序を表し、 $\theta[k]=i$ の場合、 x_i は入力変数の k 番目に位置することを表す。

[0158] (2) 特性関数二分決定グラフの節点テーブルの作成

変数 X, Y の初期順序 P の決定がされた後、BDD生成手段7は、決定された初期順序 P に従って、論理仕様記憶手段2に記憶された論理仕様に従って、特性関数二分決定グラフの節点テーブルを生成する。この節点テーブルは、節点テーブル記憶手段8に格納される。尚、論理仕様から特性関数二分決定グラフを生成するアルゴリズムに関しては、すでに種々のアルゴリズムが公知であるため、ここでは説明を省略する。

[0159] 図13は節点テーブルのデータ構造を表す図である。1つの節点に対応する節点データは、(変数ラベル, 0枝の子節点のアドレス, 1枝の子節点のアドレス)の3つのデータ組からなる。変数ラベルには、各節点に対応する入力変数又は出力変数を特定する符号が格納される。通常は、計算処理の便宜のため、 $m+n$ 個の各変数 $x_1, \dots, x_n, y_0, \dots, y_{m-1}$ には2進数の符号が割り当てられるが、ここでは説明の都合上、変数 $x_1, \dots, x_n, y_0, \dots, y_{m-1}$ の記号をそのまま使用する。「0枝」(0-edge)とは、その節点に対応する変数の値(リテラル)が0であるときの節点間の遷移を表す枝(edge)をいい、「1枝」(1-edge)とは、その節点に対応する変数の値(リテラル)が1であるときの節点間の遷移を表す枝をいう。「0枝の子節点」(「1枝の子節点」)とは、その節点に対して0枝(1枝)に沿って遷移した先の節点をいう。「節点のアドレス」とは、節点テーブル記憶手段8において、その節点データが格納されている記憶装置の論理アドレスをいう。

[0160] 従って、ある節点に対応する変数の値が与えられた場合に、その変数値に対する遷移先の節点を参照する場合には、変数値0, 1に応じて0枝の子節点のアドレス, 1枝の子節点のアドレスを参照すれば、特性関数二分決定グラフの径路(path)を辿る

ことができる。

[0161] [例8]

入力変数 $X=(x_1, x_2, x_3)$ に対して式(31)により表される多出力論理関数 $f(X)$ について、変数順序 P は、上記アルゴリズムによって $P=(x_1, x_2, y_0, x_3, y_1)$ となる。

[0162] [数31]

$$\begin{aligned} F(X) &= (f_0(X), f_1(X)) \\ &= (x_1 x_2, x_1 \vee x_3) \end{aligned} \quad (31)$$

[0163] ここで、式(31)の多出力論理関数 $f(X)$ の特性関数二分決定グラフは、図14(b)により表される。従って、図14(b)の特性関数二分決定グラフに対応する節点テーブルは、図14(a)のようになる。ここで、終端節点のうち、特性関数値0に対応する終端節点にはアドレス0、特性関数値1に対応する終端節点にはアドレス1が割り当てられる。

[例終わり]

[0164] (3) 変数 X, Y の順序の最適化

次に、変数 X, Y が初期順序により順序づけられた特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段8に格納されると、変数順序最適化手段9は、変数 X, Y の間の順序の入れ替えを実行し、変数 X, Y 間の順序の最適化を行う。このとき、変数 X の順序の入れ替えには式(32)の条件(すなわち、「集合 $\text{supp}(f_j)$ に属するすべての入力変数 x_i は y_j よりも高位である。」という条件)が課される。

[0165] [数32]

$$x_i \succ y_j \quad (\forall x_i \in \text{supp}(f_j)) \quad (32)$$

(但し、 $z_p \succ z_q$ は、変数 z_p の高さが変数 z_q の高さよりも高い(z_p は z_q より高位である)ことを表す。)

[0166] [例9]

式(31)の多出力論理関数 $F(X)$ で表現される論理仕様に対して、上記の変数順序 $P=(x_1, x_2, y_0, x_3, y_1)$ による特性関数二分決定グラフの節点テーブルが、図14のように生成され、節点テーブル記憶手段8に格納されているとする。変数順序最適化手段9は、この節点テーブルに対して、上記条件のもとでの変数順序 P の入れ替えを行う。そ

して、入れ替え後の変数順序P'により、特性関数二分決定グラフの再構築を行い、その結果、特性関数二分決定グラフの幅の総和が小さくなった場合にのみ、変数順序P'の節点テーブルで節点テーブル記憶手段8の内容を更新する。

[0167] 例えば、変数順序最適化手段9が上記の変数順序Pを順序 $P'=(x_1, x_2, x_3, y_0, y_1)$ のように入れ替えを行った場合、特性関数二分決定グラフは図15(b)のようになる。また、この特性関数二分決定グラフに対応する節点テーブルは図15(b)のようになる。図15(a)の節点テーブルの大きさと図14(a)の節点テーブルの大きさは同じである。従って、この場合、節点テーブル記憶手段8の内容の更新は行われない。

[0168] このような変数順序の入れ替えによる節点テーブルの再構築を総当たり(又は、適当な条件を課した試行)で試みることにより、変数順序の最適化が行われる。尚、式(31)の多出力論理関数の場合、図14の特性関数二分決定グラフが最小であるため、節点テーブル記憶手段8内の節点テーブルの更新は行われない。

[例終わり]

[0169] (4) LUTカスケード論理回路の合成

以上のように、最適化された特性関数二分決定グラフの節点テーブルが節点テーブル記憶手段8に格納されると、続いてLUTカスケード論理回路の合成処理が行われる。そこで、次に、LUTカスケード論理回路の合成処理について説明する。

[0170] 図16はLUTカスケード論理回路の合成処理の流れを表すフローチャートである。論理回路合成装置1は、分割線の高さを表す内部変数 i ($i-1$ が分割線の高さを表す。)を $n+m+1$ (n は入力変数 X の大きさ($n=|X|$)、 m は出力変数 Y の大きさ($m=|Y|$ 。))とする。また、中間変数の集合 H を空集合 ϕ とする。また、分割線より高位の変数の集合 Z_a を空集合 ϕ とする。また、関数分解を行う特性関数二分決定グラフ $B_{CF}^{current}$ を関数 f の特性関数二分決定グラフ $B_{CF}(f)$ とし、 $B_{CF}^{current}$ に含まれる変数の集合 Z_i を全変数の集合 $Z(=X \cup Y)$ に初期化する(S30)。

[0171] 次に、分割線設定手段10は、変数 i を1だけ減少する(S31)。すなわち、分割線の高さを1だけ下げる。そして、集合 Z_{temp} を $Z_a \cup \{z_i\}$ とする(S32)。ここで、 $\{z_i\}$ は高さ i の変数 z_i の集合を表す。すなわち、分割線を下げたので、集合 $\{z_i\}$ を分割線よりも高位の変数の集合 Z_{temp} に追加する。

[0172] 次に、分割線設定手段10は、関数分解可能性を判定する(S33)。関数分解の可能性の判定は、分割線で分割した場合の列複雑度 μ 及び集合 Z_{temp} の要素の数とLUTの入力数の最大値 k を比較することにより行われる。すなわち、関数分解可能な条件は、式(33)で表される。従って、式(33)の条件を満たすか否かで関数分解の可能性が判定される。

[0173] [数33]

$$\lceil \log_2 \mu \rceil < k \wedge |Z_{temp}| \leq k \quad (33)$$

[0174] 上記ステップS34の判定の結果、関数分解が可能であれば、次に、変数 i が1であるか否かを判定する(S35)。 $i=1$ ならば、これ以上関数分解はできないので、LUT生成手段14は関数 $f(Z_{temp})$ のLUTを生成し、LUT記憶手段16に格納し、終了する(S36)。また、ステップS35において、分割線の高さ i が1よりも大きい場合には、分割線設定手段10は、集合 Z_a を集合 Z_{temp} に更新して(S37)、ステップS31に戻る。

[0175] 一方、ステップS34において、関数分解が不可能であると判定される場合、分割線設定手段10は、 Z_a の大きさが $|Z_a|=|H|$ かどうかを判定する(S38)。もし、 $|Z_a|=|H|$ ならば、これ以上分割線を下げても関数分解が可能となる可能性はない。従って、この場合には、分割線設定手段10は、LUT論理回路が実現不可能である旨のメッセージを出力装置4に出力し(S39)、処理を終了する。

[0176] ステップS38において、 $|Z_a| > |H|$ であれば、短絡除去手段11は、分割線の高さ以下の高さにある変数の集合 Z_b を $Z_a - Z_b$ とする(S40)。次に、短絡除去手段11は、分割線よりも高位の部分グラフに対して短絡除去を行う。次いで、BDD幅計測手段12は、短絡除去がされた特性関数二分決定グラフについて、分割線における幅 μ_a を計測する。次に、中間変数算出手段13は、レイル数 u_a を式(34)により計算する(S41)。

[0177] [数34]

$$u_a = \lceil \log_2 \mu_a \rceil \quad (34)$$

[0178] 次に、LUT生成手段14は、 (Z_a, Z_b) を変数の分割として、 $f(Z)=g(h(Z_a), Z_b)$ の形で関数分解したときの関数 $h(Z_a)$ のLUTを生成し、LUT記憶手段16に格納する(S42)。すなわち、上記S41の短絡処理が行われていない特性関数二分決定グラフについて、

変数 Z_a に関する部分グラフを B_a 、 Z_b に関わる部分グラフを B_b とする。また、ステップS41において短絡除去がされた特性関数二分決定グラフについて、 $X_a (\subseteq Z_a)$ に関わる部分グラフを B'_a 、 Z_b に関わる部分グラフを B_b とする。LUT生成手段14は、部分グラフ B_a から、集合 Z_a に属するすべての入力変数 $x_i (\in Z_a)$ を制御入力とし、集合 Z_a に属するすべての出力変数 $y_j (\in Z_a)$ を出力するLUTを生成する。次いで、LUT生成手段14は、ステップS41において短絡除去がされた特性関数二分決定グラフについて、部分グラフ B'_a から直接接続される部分グラフ B_b 内の節点の各々に u_a ビットの符号(h_1, \dots, h_{u_a})を割り当てる。この符号(h_1, \dots, h_{u_a})を中間変数とする。そして、部分グラフ B'_a から、集合 Z_a に属するすべての入力変数 $X_a (\in Z_a)$ を制御入力とし、中間変数(h_1, \dots, h_{u_a})を出力するLUTを生成する。そして、これらのLUTをLUT記憶手段16に格納する。

[0179] 次に、BDD再構成手段15は、(Z_a, Z_b)を変数の分割として、 $f(Z) = g(h(Z_a), Z_b)$ の形で関数分解したときの関数 $g(h, Z_b)$ についての特性関数二分決定グラフを生成する(S44)。すなわち、BDD再構成手段15は、ステップS41において短絡除去がされた特性関数二分決定グラフについて、 $X_a (\subseteq Z_a)$ に関わる部分グラフを、中間変数(h_1, \dots, h_{u_a})を制御入力とする二分木に置き換える。

[0180] 次に、中間変数の集合Hを、 u_a 個の中間変数(h_1, \dots, h_{u_a})に更新する。そして、集合 Z_t を集合 Z_b と集合Hとの和集合 $Z_b \cup H$ とする(S44)。

[0181] 次に、 $|Z_t|$ がk以下かどうかを判定する(S45)。 $|Z_t| \leq k$ ならば、関数 g のLUTを生成し、LUT記憶手段16に格納し(S46)、処理を終了する。

[0182] ステップS44において、 $|Z_t| > k$ ならば、分割線の高さiを $|Z_t| + 1$ 、 $B_{CF}^{current}$ を $B_{CF}(g)$ 、集合 Z_t を集合 Z_b と集合Hとの和集合 $Z_b \cup H$ とした後(S47)、更に、集合 Z_a を集合H、関数 f を関数 g として(S48)、ステップS31に戻る。

[0183] 以上の処理により、論理関数の分解が行われ、LUTカスケード論理回路が構成される。

[0184] 以上のように、本実施例の論理回路合成装置1によれば、比較的少容量のメモリを用いてLUT論理回路の合成を実現できる。また、論理関数分解処理の演算処理速度を高速化することができる。

[0185] また、現在まで有効な論理回路合成手段が知られていなかった、多出力論理関数

の分解による途中出力を有するLUT論理回路の生成を、現実的なハードウェアを使用して短時間で実行することが可能となる。

[0186] (5) 実験結果

最後に、本発明の効果を示すため、実際に幾つかのベンチマーク関数を用いてLUTカスケード論理回路の合成を行った結果を図17に示す。実験は上記実施例1のアルゴリズムをC言語で実装し、MCNC89ベンチマーク関数に適用した。図17は、LUTの入力数 k を10に設定した場合の結果である。

[0187] 図17において、「Name」は関数名、「In」は入力数、「Out」は出力数、「LUT」は生成されるLUTの総数、「Cas」はカスケードの個数を表し、「段数」はカスケードの段数を表す。実行環境は、IBM PC/AT互換機、Pentium(登録商標)4 2.0GHz、メモリ512MByte、OSはWindows(登録商標)2000、cygwin上でgccを用いてコンパイルした。本アルゴリズムでは、出力のグループ分けを行うため、一つずつの出力をグループに加えて特性関数二分決定グラフを構成し、変数順序最適化を行う。 k が大きくなると、LUTカスケードに対応する特性関数二分決定グラフが大きくなる。LUTカスケードで構成可能な限り、グループの出力を一つずつ増やしながら変数順序最適化を行うため、大きな特性関数二分決定グラフの変数順序最適化を行う回数が増える。このため、 k が大きくなると多くの実行時間が必要となる。実行時間の殆どは、特性関数二分決定グラフの変数順序最適化のための時間である。

[0188] 本発明の効果を確認するため、非特許文献15に記載の方法との比較を行った。非特許文献15の方法は、MTBDDに基づいており、多出力論理関数の出力を幾つかのグループに分割してMTBDDで表現し、LUTカスケードで実現する。MTBDDの幅が大きすぎるため分解不可能な場合は、OR分割を用いてカスケードを分割する。図17では、非特許文献15の方法については、一つのグループの出力数は8として、出力を分割している。

[0189] 非特許文献15の方法では、実現したLUTカスケードのLUTの個数や段数にかかわらず、分割するグループの出力数を固定しているため、カスケードの個数は多くなり、段数は小さくなる傾向にある。一方、本発明に係る方法では、LUTカスケードは可能な限り、一つのグループの出力の個数を増やすので、段数は多くなり、カスケー

ドの個数は少なくなる。

実施例 2

- [0190] 図18は、本発明の実施例2に係る論理回路合成装置及びその周辺装置の構成を表す図である。図18において、論理仕様記憶手段2、入力装置3、出力装置4、出力変数順序決定手段5、全変数順序決定手段6、BDD生成手段7、節点テーブル記憶手段8、変数順序最適化手段9、分割線設定手段10、短絡除去手段11、BDD幅計測手段12、中間変数算出手段13、LUT生成手段14、BDD再構成手段15、及びLUT記憶手段16は、実施例1と同様のものである。本実施例の論理回路合成装置1'は、グラフ幅削減装置20を備えている点を特徴としている。
- [0191] 尚、本実施例においても、実施例1と同様、論理回路設計用のCAD装置において論理回路合成装置1'を使用する場合を想定している。論理回路合成装置1'は、プログラム形態として提供され、そのプログラムをコンピュータに読み込んで実行することによって、図18に示したような論理回路合成装置1'の機能構成がコンピュータで実現される。
- [0192] グラフ幅削減装置20は、節点テーブル記憶手段8に記憶された節点テーブルにより表される特性関数が不完全定義関数である場合、その特性関数二分決定グラフの幅の削減を行う。
- [0193] 図19は、図18のグラフ幅削減装置20の構成を表す図である。グラフ幅削減装置20は、BDD高さ計測手段21、分割線設定手段22、列関数生成手段23、列関数記憶手段24、両立枝生成手段25、両立グラフ記憶手段26、クリーク生成手段27、クリーク・データ記憶手段28、及び割当BDD再構成手段29を備えている。
- [0194] BDD高さ計測手段21は、節点テーブル記憶手段8に記憶されている節点テーブルによって表現される特性関数二分決定グラフの根節点の高さ t を計測する。分割線設定手段22は、高さ $t-1$ から1まで、分割の高さ lev を順次設定する。
- [0195] 列関数生成手段23は、節点データ記憶手段8に記憶された節点テーブルから、分割の高さ lev の節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する関数節点データを生成して両立グラフ記憶手段26に保存する。列関数記憶手段24は、列関数生成手段23が生成する列関数を一時的に記憶

する。

- [0196] 両立枝生成手段25は、両立グラフ記憶手段26に記憶された各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して、両立グラフ記憶手段26に記憶された関数データの更新を行う。
- [0197] クリーク生成手段27は、両立グラフ記憶手段26に記憶された両立グラフの全節点について、完全部分グラフ(クリーク)による節点被覆を行う。そして、クリークに含まれる関数節点集合であるクリーク・データを生成し、クリーク・データ記憶手段28に記憶する。
- [0198] 割当BDD再構成手段29は、クリーク・データ記憶手段28に記憶された各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数を生成する。そして、当該クリーク・データに含まれる各関数節点の列関数を、割当列関数に置換することによって、特性関数二分決定グラフの再構成を行い、節点テーブル記憶手段8の節点テーブルの更新を行う。尚、割当BDD再構成手段29は、各関数節点に対応する列関数を参照する際、節点テーブル記憶手段8の節点テーブルから必要に応じて列関数を合成するようにしてもよいが、列関数記憶手段24に一時保存されている列関数を参照するようにしてもよい。
- [0199] 以上のように構成された本実施例に係る論理回路合成装置1'について、以下その演算処理動作を説明する。
- [0200] 全体の流れとしては、最初に、図11のステップS1〜S3の処理により、節点データ記憶手段8に、不完全定義関数の特性関数二分決定グラフの節点テーブルが保存される。次に、グラフ幅削減装置20による特性関数二分決定グラフの幅削減処理が実行される。そして、最後に、図11のステップS4のLUT論理回路の合成が行われる。ここで、本実施例の論理回路合成装置1'は、グラフ幅削減装置20による特性関数二分決定グラフの幅削減処理以外の演算処理は、実施例1と同様である。そこで、ここではグラフ幅削減装置20の演算処理についてのみ説明する。
- [0201] 図20は、グラフ幅削減装置20のグラフ幅削減処理の全体の流れを表すフローチャ

ートである。最初に、BDD高さ計測手段21が、節点テーブル記憶手段8に記憶された節点テーブルを探索して、特性関数二分決定グラフの根節点の高さ t を決定する。根節点の高さ t が決定されると、分割線設定手段22は、分割の高さ lev を $t-1$ に設定する(S50)。

[0202] 次に、列関数生成手段23は、分割の高さ lev におけるすべての列関数の集合を作成する(S51)。生成された列関数に対しては、関数節点の節点ラベルが付与され、両立グラフ記憶手段26に関数節点データとして保存される。また、特性関数二分決定グラフの大きさが小さい場合には、計算速度を向上させるために、列関数生成手段23が生成した各列関数を、列関数記憶手段24に保存するようにしてもよい。

[0203] 次に、両立枝生成手段25は、列関数生成手段23が生成する各列関数から、両立する列関数の組を選出する。そして、両立グラフ記憶手段26に記憶された両立する列関数に対する関数節点データに、それらの関数節点同士を連結する両立枝を追加する。両立枝が追加された関数節点データは、両立グラフ記憶手段26に保存され、関数節点データの更新が行われる(S52)。この処理は、列関数生成手段23が生成するすべての列関数に対して、すべての2つの列関数組の組み合わせについて実行される。この処理によって、両立グラフ記憶手段26に、両立グラフを表現する関数節点データのテーブルが生成される。

[0204] 次に、クリーク生成手段27は、両立グラフ記憶手段26に記憶された関数節点データのテーブルが表す両立グラフに対して、クリーク集合による節点被覆処理を実行する(S53)。この処理の詳細については後述するが、これによって、両立グラフのすべての節点について、完全部分グラフ(クリーク)による節点被覆が行われ、クリーク・データが生成される。生成されたクリーク・データは、クリーク・データ記憶手段28に保存される。

[0205] 尚、「完全部分グラフ(クリーク)」とは、両立グラフの部分グラフであって、当該部分グラフの任意の一つの節点が他のすべての節点と枝により連結されているものをいう。また、クリーク・データとは、クリークに含まれる関数節点の集合のデータをいう。

[0206] 次に、割当BDD再構成手段29は、節点テーブル記憶手段8に記憶された節点テーブルに基づき、クリーク・データ記憶手段28に記憶された一つのクリーク・データに

ついて、当該クリーク・データに含まれる関数節点に対応する列関数を生成し、列関数記憶手段24に一時的に保存する。そして、当該クリーク・データに属するすべての関数節点に対応する列関数について、ドント・ケアへの定数の割り当てを行い、同一の割当列関数を生成する。そして、当該クリーク・データに属するすべての関数節点に対応する列関数を、生成した割当列関数により置換した特性関数二分決定グラフとなるように、節点テーブル記憶手段8内の節点テーブルの書き換えを行う。この割当列関数を用いた節点テーブルの書き換え処理を、すべてのクリーク・データに対して実行することにより、特性関数二分決定グラフの再構成を行う(S54)。

[0207] 次に、分割線設定手段22は、分割の高さlevを1だけ減少させる(S55)。そして、分割の高さlevが1以上であれば(S56)、ステップS51に戻る。分割の高さlevが0となった場合には(S56)、グラフ幅削減処理を終了する。

[0208] 以上のようなグラフ幅削減処理によって、上述したアルゴリズム3が実行され、効果的な特性関数二分決定グラフの幅の削減が実行される。

[0209] 次に、上記ステップS53におけるクリーク集合による両立グラフの節点被覆処理について説明する。図21は、クリーク集合による両立グラフの節点被覆処理の流れを表すフローチャートである。

[0210] クリーク生成手段27は、内部変数として、クリーク・データのリストを表す変数C(以下「集合C」という。)、クリーク被覆が行われていない関数節点のリストを表す変数 S_a (以下「節点集合 S_a 」という。)、一つのクリークに含まれる関数節点のリストを表す変数 S_i (以下「節点集合 S_i 」という。)、及びクリーク変数に追加する関数節点の候補のリストを表す変数 S_b (以下「節点集合 S_b 」という。)を有する。

[0211] まず、クリーク生成手段27は、クリーク・データの集合Cを空集合に初期化する(S60)。そして、両立グラフのすべての関数節点を節点集合 S_a に追加して初期化を行った後、節点集合 S_a から枝を持たない関数節点を除去するとともに、これらの関数節点を、要素1のクリーク集合として、集合Cの要素に追加する(S61)。

[0212] 次に、クリーク生成手段27は、節点集合 S_a が空集合でないか否かを判定する(S62)。

[0213] 節点集合 S_a が空集合でない場合、クリーク生成手段27は、節点集合 S_a の中で枝数

が最小である関数節点を探索し、これを関数節点 v_i とする(S63)。次に、クリーク生成手段27は、節点集合 S_i を $\{v_i\}$ に初期化する(S64)。そして、節点集合 S_a の中で関数節点 v_i に接続している関数節点を探索し、これを節点集合 S_b とする(S65)。

[0214] 次に、クリーク生成手段27は、節点集合 S_b が空集合か否かを判定する(S66)。節点集合 S_b が空集合でなければ、節点集合 S_b に属する関数節点の中で、枝数が最小のものを探索し、これを関数節点 v_j とする。そして、節点集合 S_i に $\{v_j\}$ を追加するとともに、節点集合 S_b から関数節点 v_j を除去する(S67)。そして、節点集合 S_b に属する関数節点のうち、関数節点 v_j に接続していない関数節点を節点集合 S_b から除去する(S68)。そして、ステップS66に戻る。

[0215] ステップS66において、節点集合 S_b が空集合の場合、集合Cの要素に節点集合 S_i をクリークとして追加するとともに、未被覆の節点集合 S_a から節点集合 S_i を除去する(S69)。そして、ステップS62に戻る。

[0216] ステップS62において、未被覆の節点集合 S_a が空集合となった時点で、集合Cをクリーク・データとしてクリーク・データ記憶手段28に保存し、クリーク集合による両立グラフの節点被覆処理を終了する。

[0217] 以上のような処理によって、両立グラフのすべての関数節点が、クリークにより被覆され、クリーク・データが生成される。

[0218] [例10]

真理値が例6の(表4)により表される4入力2出力不完全定義関数について、特性関数二分決定グラフの幅を削減する。(表4)により表される4入力2出力不完全定義関数の特性関数二分決定グラフは、図4(a)で表される。特性関数二分決定グラフの高さ t は6である。これを、本実施例のグラフ幅削減装置20によって特性関数二分決定グラフの幅の削減処理を行う。

[0219] まず、最初に分割の高さ lev は $t-1=5$ 、すなわち、変数 x_2 の高さに設定される。 $lev=5$ において、分解表は(表5)に示したようになる。(表5)の列関数は両立しないため、ドント・ケアへの定数の割り当てはできない。

[0220] [表5]

$$X_1 = \{x_1\}$$

	0	1
000	d1	01
001	d1	01
010	00	10
011	00	10
100	dd	1d
101	dd	1d
110	10	d0
111	11	d1
	Φ_1	Φ_2

$$X_2 = \{x_2, x_3, x_4\}$$

[0221] 次に、分割の高さをlev=4とする(図22(a)参照)。lev=4の分解表は、(表6)のようになる。

[0222] [表6]

$$X_1 = \{x_1\}$$

	00	01	10	11
00	d1	dd	01	1d
01	d1	dd	01	1d
10	00	10	10	d0
11	00	11	10	d1
	① Φ_1	② Φ_2	③ Φ_3	④ Φ_4

$$X_2 = \{x_2, x_3, x_4\}$$

[0223] (表6)より、4つの列関数 $\Phi_1 \sim \Phi_4$ が得られる。列関数 Φ_2 と列関数 Φ_4 は両立する。従って、両立グラフは、図22(b)のように表される。図22(b)の両立グラフについて、クリークによる節点被覆処理を行うと、図22(b)の点線で示した3つのクリーク S_1, S_2, S_3 が生成される。そこで、クリーク S_3 に含まれる2つの列関数 Φ_2, Φ_4 に対して定数の割り当てを行うと、(表7)のようになる。これらの列関数から特性関数二分決定グラフを生成すると、図23(a)のようになる。図22(a)のグラフの節点2, 4は、図23(a)では、節点11に置き換えられている。

[0224] [表7]

		$X_1 = \{x_1\}$			
		00	01	10	11
$X_2 = \{x_2, x_3, x_4\}$	00	d1	1d	01	1d
	01	d1	1d	01	1d
	10	00	10	10	10
	11	00	11	10	11
		① Φ_1	② Φ'_2	③ Φ_3	④ Φ'_4

[0225] 次に、分割の高さをlev=3(変数 y_1 の高さ)とする(図23(a)参照)。図23(a)より、分割の高さをlev=3を横切る各枝に対応して、6つの列関数 $\Phi_1 \sim \Phi_6$ が得られる。図23(a)では、各列関数 $\Phi_1 \sim \Phi_6$ の添字に対応して、それぞれの枝に番号が付されている。列関数 Φ_1, Φ_5 、列関数 Φ_3, Φ_4, Φ_6 は両立する。従って、両立グラフは、図23(b)のように表される。図23(b)の両立グラフについて、クリークによる節点被覆処理を行うと、例えば、図23(b)の点線で示した4つのクリーク S_1, S_2, S_3, S_4 が生成される。そこで、クリーク S_1 に含まれる2つの列関数 Φ_1, Φ_5 に対して定数の割り当てを行い、クリーク S_2 に含まれる2つの列関数 Φ_3, Φ_4 に対して定数の割り当てを行う。そして、生成された割当列関数を用いて、特性関数二分決定グラフの再構成を行うと、グラフ構造は図24のようになる。図23(a)の節点6, 8と節点7, 10を、図24では、それぞれ節点12, 13で置き換えている。

[0226] 分割の高さをlev=2, 1では、各列関数は両立しないので、これ以上特性関数二分決定グラフの再構成は行われない。

[0227] 図22(a)と図24とを比較すると、グラフ幅削減処理の結果、最大幅は8から4に削減され、非終端節点数は15から12に削減されたことが分かる。これを、上述したアルゴリズム1の結果と比較すると、アルゴリズム1を適用した結果(図4(b)参照)に比べ、最大幅がより削減されていることが分かる。

[例終わり]

[0228] 最後に、幾つかのベンチマーク関数を用いて、本実施例に係るグラフ幅削減装置1'の性能を試験した結果について説明する。(表8)は、各ベンチマーク関数について、本実施例の論理回路合成装置1'によりLUTカスケード論理回路の合成を行った

結果を表す。

[0229] [表8]

Name	DC = 0			Alg.3		
	# Cells	# LUTs	# Cas	# Cells	# LUTs	# Cas
5-7-11-13 RNS	6	35	3	4	29	2
7-11-13-17 RNS	9	53	3	8	52	3
11-13-15-17 RNS	19	118	5	18	108	5
4-digit 11-nary to binary	4	29	2	4	28	2
4-digit 13-nary to binary	4	31	2	4	30	2
5-digit decimal to binary	8	50	3	7	48	2
6-digit 5-nary to binary	6	44	2	5	36	2
6-digit 6-nary to binary	5	38	2	5	29	2
6-digit 7-nary to binary	10	68	3	9	58	3
10-digit ternary to binary	9	59	3	6	48	2
3-digit decimal adder	7	37	2	3	17	1
4-digit decimal adder	10	64	2	4	24	1
2-digit decimal multiplier	8	51	3	7	48	3
Total	105	677	35	84	555	30
Ratio	1.00	1.00	1.00	0.80	0.82	0.86

[0230] それぞれのベンチマーク関数は以下の通りである。

[0231] (1) 剰余数表現 (Residue Number System, RNS) を2進表現に変換する回路 (Residue Number to Binary Number Converters)

• 5-7-11-13RNS (14-input 13-output, DC 69.5%: 4-digit RNS number, where the moduli are 5, 7, 11, and 13)

• 7-11-13-17 RNS (16-input 15-output, DC 74.0%)

• 11-13-15-17 RNS (17-input 16-output, DC 72.2%)

[0232] (2) i桁のk進数を2進数に変換する回路 (k-nary to Binary Converters)

• 4-digit 11-nary to binary (16-input 14-output, DC 77.7%)

• 4-digit 13-nary to binary (16-input 15-output, DC 56.4%)

• 5-digit decimal to binary (20-input 17-output, DC 90.5%)

• 6-digit 5-nary to binary (18-input 14-output, DC 94.0%)

• 6-digit 6-nary to binary (18-input 16-output, DC 82.2%)

• 6-digit 7-nary to binary (18-input 17-output, DC 55.1%)

- 10-digit ternary to binary (20-input 16-output, DC 94.4%)

[0233] (3) i桁の10進数の加算回路、乗算回路 (Decimal Adders and a Multiplier)

- 3-digit decimal adder (24-input 16-output, DC 94.0%)
- 4-digit decimal adder (32-input 20-output, DC 97.7%)
- 2-digit decimal multiplier (16-input 16-output, DC 84.7%)

[0234] (表8)において、「DC=0」は、すべてのドント・ケアに0を割り当てた場合を示し、「Alg.3」は、本実施例に係る論理回路合成装置1'によりアルゴリズム3を適用した場合を示している。「# Cells」の欄は、カスケード内のセル数を表す。「# LUTs」の欄は、LUTの出力数の総和を表す。「# Cas」の欄は、カスケードの数を表す。(表8)より、本実施例に係る論理回路合成装置1'により、カスケード内のセル数とLUTの出力数の総和を効果的に削減することが可能であることが分かる。

[0235] 例えば、5-7-11-13 RNSを考える。この関数は、入力の組み合わせに対し69.5%のドント・ケアを含む。この関数についてLUTカスケード論理回路を合成する。条件として、各セルの入力の本数は制限されており、入力数が最大12まで許されるものとする。

[0236] ドント・ケアに0を割り当ててLUTカスケード論理回路を合成すると、図25(a)のようになる。これに対し、本実施例に係る論理回路合成装置1'によりLUTカスケード論理回路を合成すると、図25(b)のようになる。明らかに、本実施例に係る論理回路合成装置1'によりLUTカスケード論理回路の回路規模を減少させることが可能であることが分かる。

産業上の利用可能性

[0237] 本発明は、LUT型の論理回路を自動設計する論理回路設計用のCAD装置、LUT型FPGAの論理合成ツール等に適用することができる。

請求の範囲

- [1] 入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(1)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフ(Binary Decision Diagram for Characteristic Function : BDD for CF)の幅を削減するグラフ幅削減装置であって、以下の構成を備えているグラフ幅削減装置:

多出力論理関数 $F(X)$ の特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルを記憶する節点テーブル記憶手段;

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定手段;

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定手段により設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成手段;

及び、前記列関数生成手段が生成する各列関数のうち、両立する列関数に対してドント・ケアに定数を割り当てることによって同一の列関数(以下「割当列関数」という。)とするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当BDD再構成手段。

[数1]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{ \bar{y}_i f_{i,0} \vee y_i f_{i,1} \vee f_{i,d} \} \quad (1)$$

但し、 $f_{i,0}$, $f_{i,1}$, $f_{i,d}$ は、それぞれ、式(2)で定義されるOFF関数、ON関数、及びDC関

数を表す。

[数2]

$$f_{i,0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (2)$$

- [2] 前記列関数を節点(以下「関数節点」という。)とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝(以下「両立枝」という。)により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ(以下「関数節点データ」という。)のテーブルとして記憶するための両立グラフ記憶手段;

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成手段;

及び、前記両立グラフのすべての節点について、完全部分グラフ(以下「クリーク」という。)による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリーク・データを生成するクリーク生成手段;

を備え、

前記列関数生成手段は、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定手段により設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するものであり、

前記割当BDD再構成手段は、前記クリーク生成手段が生成する前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新するものであること

を特徴とする請求項1記載のグラフ幅削減装置。

- [3] 前記分割線設定手段は、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから下位に向かって順次分割の高さlevの設定を行うものであり、

前記割当BDD再構成手段は、前記分割線設定手段が設定する前記各分割の高さlevにおいて逐次前記特性関数二分決定グラフ再構成を行うことを特徴とする請求項1又は2記載のグラフ幅削減装置。

- [4] 入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とする多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ の特性関数二分決定グラフから、前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブル(Look-Up Table : LUT)を生成する論理回路合成装置であって、以下の構成を備えている論理回路合成装置：

完全定義関数である前記多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(3)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i), e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段；

前記ルックアップ・テーブルを記憶するLUT記憶手段；

前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定手段；

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 y_r ($y_r \in Y$)に関わる節点 v_j 及びその親節点 v_k の節点データについて、当該節点 v_j の枝 $e_0(v_j), e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k), e_1(v_k)$ のうち当該節点 v_j を特定する枝 $e_c(v_k)$ を、当該節点 v_j の前記枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を行う短絡除去手段；

前記短絡除去手段により短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し(但し、同じ節点を特定するものは1つと数え、定数0に向かう枝は無視する。)、その個数を前記分割の高さlevの分割線における幅Wとして出力するBDD幅計測手段;

前記BDD幅計測手段が出力する幅Wに基づき、式(4)の演算により中間変数の個数uを算出する中間変数算出手段;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納するLUT生成手段;

及び、前記中間変数算出手段が算出する前記中間変数の個数uと等しい制御入力数を有する二分木(binary tree)を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するBDD再構成手段。

[数3]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X)) \quad (3)$$

[数4]

$$u = \lceil \log_2 W \rceil \quad (4)$$

- [5] 前記節点テーブル記憶手段には、出力にドント・ケアを含む不完全定義関数である前記多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(5)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表

現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶されていること

を特徴とする請求項4記載の論理回路合成装置。

[数5]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{\bar{y}_i f_{i,0} \vee y_i f_{i,1} \vee f_{i,d}\} \quad (5)$$

但し、 $f_{i,0}$, $f_{i,1}$, $f_{i,d}$ は、それぞれ、式(6)で定義されるOFF関数、ON関数、及びDC関数を表す。

[数6]

$$f_{i,0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (6)$$

- [6] 請求項1乃至3の何れか一記載のグラフ幅削減装置を備え、
前記短絡除去手段は、前記グラフ幅削減装置によって前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を削減し更新された前記節点テーブルについて短絡除去処理を行うものであることを特徴とする請求項5記載の論理関数合成装置。

- [7] 前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f が i 番目であることを表す。)とし、論理関数 $f_j(\in F(X))$ が依存する入力変数の集合を $\text{supp}(f_j)$ としたとき、式(7)で表される T の値が最小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定手段；

前記各入力変数 x_i ($\in X$) 及び出力を表す変数 y_j ($\in Y$) の順序を、式(8)を満たす順序 P に決定する全変数順序決定手段；

及び、前記全変数順序決定手段で決定された順序 P に従って、特性関数二分決定

グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成手段
;

を備えていることを特徴とする請求項4乃至6の何れか一記載の論理回路合成装置。

[数7]

$$T = \sum_{k=0}^{m-1} \left| \bigcup_{l=0}^k \text{supp}(f_{\pi[l]}) \right| \quad (7)$$

[数8]

$$P = \left(\text{supp}(f_{\pi[0]}), y_{\pi[0]}, \text{supp}(f_{\pi[1]}) - \text{supp}(f_{\pi[0]}), y_{\pi[1]}, \text{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^1 \text{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \right. \\ \left. \dots, \text{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \text{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \quad (8)$$

- [8] 入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(9)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段を備えたシステムにおいて、当該特性関数二分決定グラフの幅を削減するグラフ幅削減方法であって、以下の各ステップを有するグラフ幅削減方法:

前記節点テーブルによって表現される特性関数二分決定グラフを分割する高さ(以下「分割の高さ」という。)levの設定を行う分割線設定ステップ;

前記節点データ記憶手段に記憶された前記節点テーブルから、前記特性関数二分決定グラフを前記分割線設定ステップで設定された前記分割の高さlevで分割し関数分解することによって得られる分解表の列を表す関数である列関数を生成する列関数生成ステップ;

及び、前記列関数生成ステップにおいて生成される各列関数のうち、両立する列

関数に対してドント・ケアに定数を割り当てることによって同一の列関数(以下「割当列関数」という。)にするとともに、これらの新たな割当列関数を用いて前記特性関数二分決定グラフ再構成し前記節点テーブル記憶手段の節点テーブルを更新する割当BDD再構成ステップ。

[数9]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{\bar{y}_i f_{i,0} \vee y_i f_{i,1} \vee f_{i,d}\} \quad (9)$$

但し、 $f_{i,0}$, $f_{i,1}$, $f_{i,d}$ は、それぞれ、式(10)で定義されるOFF関数、ON関数、及びDC関数を表す。

[数10]

$$f_{i,0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (10)$$

- [9] 前記システムは、前記列関数を節点(以下「関数節点」という。)とするグラフであって、互いに両立する複数の列関数に対応する関数節点同士が枝(以下「両立枝」という。)により連結されたグラフである両立グラフを、前記各関数節点の列関数ラベル及び当該関数節点に連結する両立枝のデータ(以下「関数節点データ」という。)のテーブルとして記憶するための両立グラフ記憶手段を備えており、

前記列関数生成ステップにおいては、前記節点データ記憶手段に記憶された前記節点テーブルから、前記分割線設定ステップで設定された前記分割の高さlevの節点の各枝に対応する列関数を生成し、それら各列関数に対応した列関数ラベルを有する前記関数節点データを生成して前記両立グラフ記憶手段に保存するとともに、

前記両立グラフ記憶手段に記憶された前記各関数節点データに対する列関数から、両立する列関数の組を選出し、それら両立する列関数に対する関数節点データにそれらの関数節点同士を連結する両立枝を追加して前記両立グラフ記憶手段に記憶された関数節点データの更新を行う両立枝生成ステップ;

及び、前記両立グラフのすべての節点について、完全部分グラフ(以下「クリーク」という。)による節点被覆を行うことにより、クリークに含まれる関数節点集合であるクリー

ク・データを生成するクリーク生成ステップ;

を有し、

前記割当BDD再構成ステップにおいては、前記クリーク生成ステップにおいて生成される前記各クリーク・データに対して、当該クリーク・データに含まれる各関数節点に対応する列関数のドント・ケアに定数を割り当てて同一の割当列関数とすることにより、前記特性関数二分決定グラフを再構成し前記節点テーブル記憶手段の前記節点テーブルを更新すること

を特徴とする請求項8記載のグラフ幅削減方法。

- [10] 前記分割線設定ステップにおける前記分割の高さlevを、前記節点テーブル記憶手段に記憶された前記節点テーブルにより表現される特性関数二分決定グラフの根節点の子の節点の高さから最下位の高さまで順次変更しながら、前記分割線設定ステップ乃至前記割当BDD再構成ステップを実行すること
を特徴とする請求項8又は9記載のグラフ幅削減方法。

- [11] 入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とする完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(11)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2$, $m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i)$, $e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶された節点テーブル記憶手段;

及び、ルックアップ・テーブル(Look-Up Table : LUT)を記憶するためのLUT記憶手段;

を備えたシステムにおいて、

前記特性関数二分決定グラフから、前記多出力論理関数 $F(X)$ に対応する論理回路を構成するためのデータであるルックアップ・テーブルを合成する論理回路合成方法であって、以下のステップを有する論理回路合成方法:

前記節点テーブルにより表現される特性関数二分決定グラフを分割する高さ(以下

「分割の高さ」という。)levの設定を行う分割線設定ステップ;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものであって、出力を表す変数 $y_r (\in Y)$ に関わる節点 v_j 及びその親節点 v_k の節点データについて、当該節点 v_j の枝 $e_0(v_j), e_1(v_j)$ の一方 $e_a(v_j)$ が $\chi(X, Y)=0$ に関わる終端節点を特定する場合、当該節点 v_j の親節点 v_k の枝 $e_0(v_k), e_1(v_k)$ のうち当該節点 v_j を特定する枝 $e_c(v_k)$ を、当該節点 v_j の前記枝 $e_a(v_j)$ 以外の枝 $e_b(v_j)$ に置き換える短絡除去処理を実行する短絡除去ステップ;

前記短絡除去処理がされた特性関数二分決定グラフの各非終端節点であって前記分割の高さlevより高い高さにある非終端節点に属する枝のうち、前記分割の高さlevより低い高さの非終端節点の子節点を特定するものの個数を計数し(但し、同じ節点を特定するものは1つと数え、定数0に向かう枝は無視する。)、その個数を前記分割の高さlevの分割線における幅Wとして出力するBDD幅計測ステップ;

前記幅Wに基づき、式(12)の演算により中間変数の個数uを算出する中間変数算出ステップ;

前記節点テーブル記憶手段に格納された非終端節点の節点データのうち、前記特性関数二分決定グラフを前記分割の高さlevの分割線において2つの部分グラフ B_0, B_1 に分割したときに根節点を含む側の部分グラフ B_0 に属するものについて、ルックアップ・テーブルを生成しそれをLUT記憶手段に格納するLUT生成ステップ;

及び、前記中間変数算出ステップにおいて算出された前記中間変数の個数uと等しい制御入力数を有する二分木(binary tree)を生成するとともに、前記節点テーブル記憶手段に格納されている特性関数二分決定グラフの部分グラフ B_0 に属する非終端節点の節点データを、前記二分木を表す節点データで置き換えることにより、特性関数二分決定グラフを再構成し、この再構成された特性関数二分決定グラフの各非終端節点の節点データにより前記節点テーブル記憶手段に格納された節点テーブルを更新するBDD再構成ステップ。

[数11]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} (y_i \equiv f_i(X)) \quad (11)$$

[数12]

$$u = \lceil \log_2 W \rceil \quad (12)$$

- [12] 前記節点テーブル記憶手段には、入力変数を $X=(x_1, \dots, x_n)$ ($n \in \text{自然数}$)とし出力にドント・ケアを含む不完全定義関数である多出力論理関数 $F(X)=(f_0(X), \dots, f_{m-1}(X))$ に対して式(13)により定義される特性関数 $\chi(X, Y)$ (但し、 $Y=(y_0, \dots, y_{m-1})$ ($m \geq 2, m \in \text{自然数}$)は $F(X)$ の出力を表す変数。)を表現する特性関数二分決定グラフが、当該特性関数二分決定グラフのそれぞれの非終端節点 v_i について、当該非終端節点に関わる変数 z_i ($z_i \in (X \cup Y)$)に付与される変数ラベル並びに当該変数 z_i ($z_i \in (X \cup Y)$)の値が0のとき及び1のときに遷移する先の子節点を特定する一対の枝 $e_0(v_i), e_1(v_i)$ からなる節点データのテーブルである節点テーブルとして記憶されていることを特徴とする請求項11記載の論理回路合成方法。

[数13]

$$\chi(X, Y) = \bigwedge_{i=0}^{m-1} \{ \bar{y}_i f_{i,0} \vee y_i f_{i,1} \vee f_{i,d} \} \quad (13)$$

但し、 $f_{i,0}, f_{i,1}, f_{i,d}$ は、それぞれ、式(14)で定義されるOFF関数、ON関数、及びDC関数を表す。

[数14]

$$f_{i,0}(X) = \begin{cases} 1 & (X \in f_i^{-1}(0)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,1}(X) = \begin{cases} 1 & (X \in f_i^{-1}(1)) \\ 0 & (\text{otherwise}) \end{cases}, f_{i,d}(X) = \begin{cases} 1 & (X \in f_i^{-1}(d)) \\ 0 & (\text{otherwise}) \end{cases} \quad (14)$$

- [13] 前記節点テーブル記憶手段に記憶された前記節点テーブルにより表される特性関数二分決定グラフの幅を、請求項8乃至10の何れかに記載のグラフ幅削減方法によって削減し、前記節点テーブル記憶手段に記憶された前記節点テーブルを更新した後に、前記分割線設定ステップ乃至前記BDD再構成ステップを実行する請求項12記載の論理回路合成方法。

- [14] 前記多出力論理関数 $F(X)$ の要素をなす論理関数 $f_0(X), \dots, f_{m-1}(X)$ の順序を $\pi = (\pi[0], \dots, \pi[m-1])$ (但し、 $\pi[i]=j$ は、 f が i 番目であることを表す。)とし、論理関数 $f_j(\in F(X))$ が依存する入力変数の集合を $\text{supp}(f_j)$ としたとき、式(15)で表される T の値が最小となるように前記多出力論理関数 $F(X)$ の要素の順序 π を決定する出力変数順序決定ステップ;

前記各入力変数 $x_i(\in X)$ 及び出力を表す変数 $y_j(\in Y)$ の順序を、式(16)を満たす順序 P に決定する全変数順序決定ステップ;

及び、前記順序 P に従って、特性関数二分決定グラフの節点データを生成し前記節点テーブル記憶手段に格納するBDD生成ステップ;

を実行した後に、前記分割線設定ステップ乃至前記BDD再構成ステップを実行することを特徴とする請求項11乃至14の何れか一記載の論理回路合成方法。

[数15]

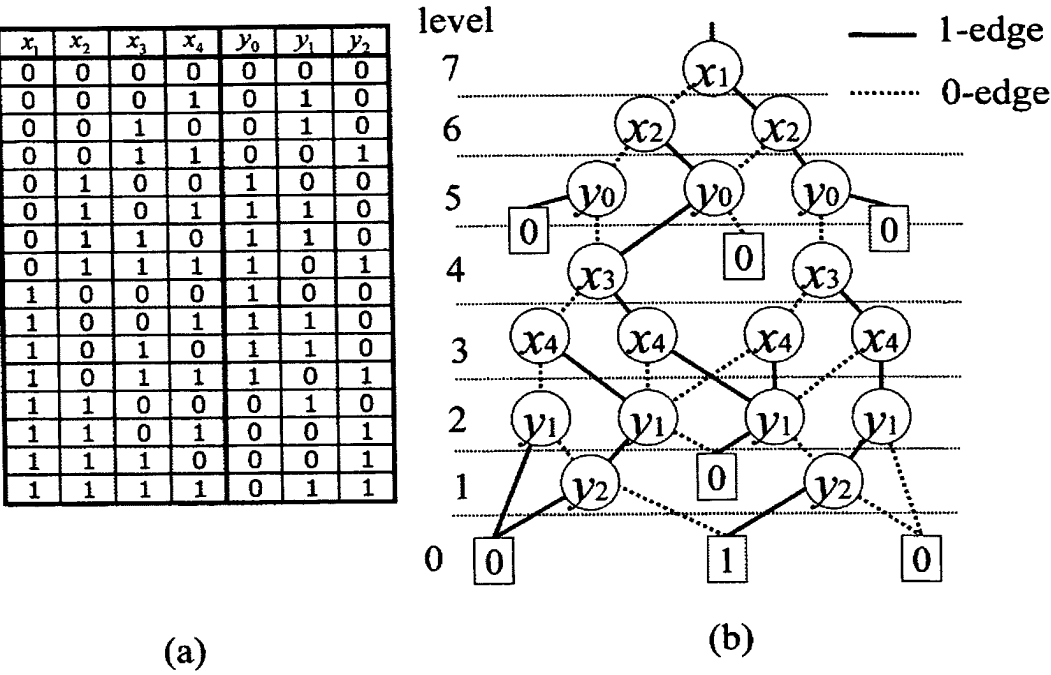
$$T = \sum_{k=0}^{m-1} \left| \bigcup_{i=0}^k \text{supp}(f_{\pi[i]}) \right| \quad (15)$$

[数16]

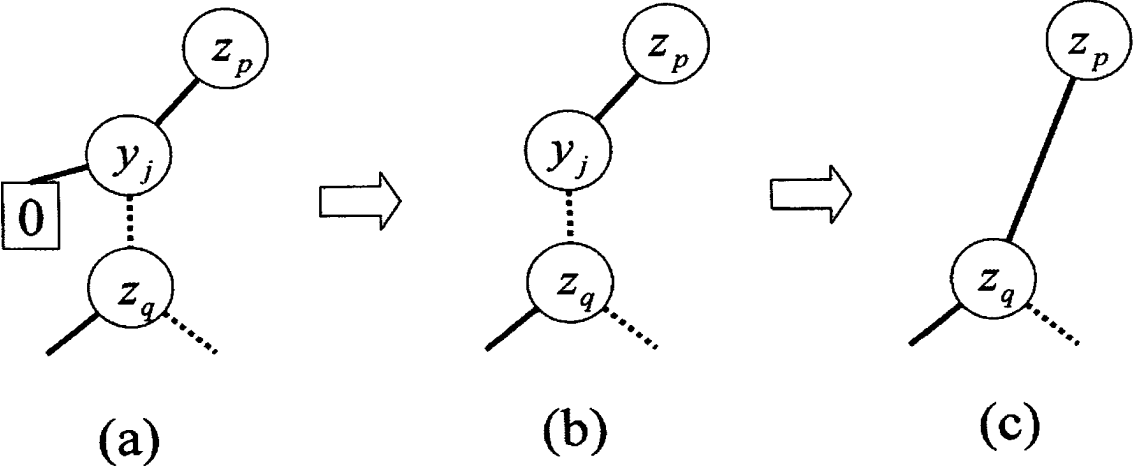
$$P = \left(\text{supp}(f_{\pi[0]}), y_{\pi[0]}, \text{supp}(f_{\pi[1]}) - \text{supp}(f_{\pi[0]}), y_{\pi[1]}, \text{supp}(f_{\pi[2]}) - \left(\sum_{k=0}^1 \text{supp}(f_{\pi[k]}) \right), y_{\pi[2]}, \dots, \text{supp}(f_{\pi[m-1]}) - \left(\sum_{k=0}^{m-2} \text{supp}(f_{\pi[k]}) \right), y_{\pi[m-1]} \right) \quad (16)$$

- [15] コンピュータに請求項8乃至10の何れかに記載のグラフ幅削減方法を実行させるプログラム。
- [16] コンピュータに請求項11乃至15に何れかに記載の論理回路合成方法を実行させるプログラム。
- [17] 請求項11乃至15に何れかに記載の論理回路合成方法により論理合成されたプログラマブル論理回路。

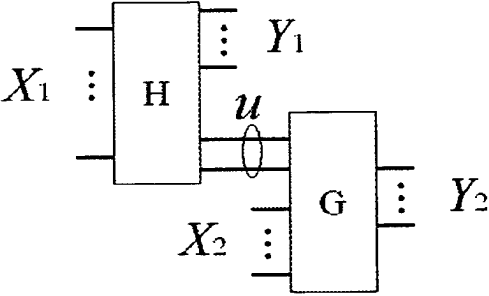
[図1]



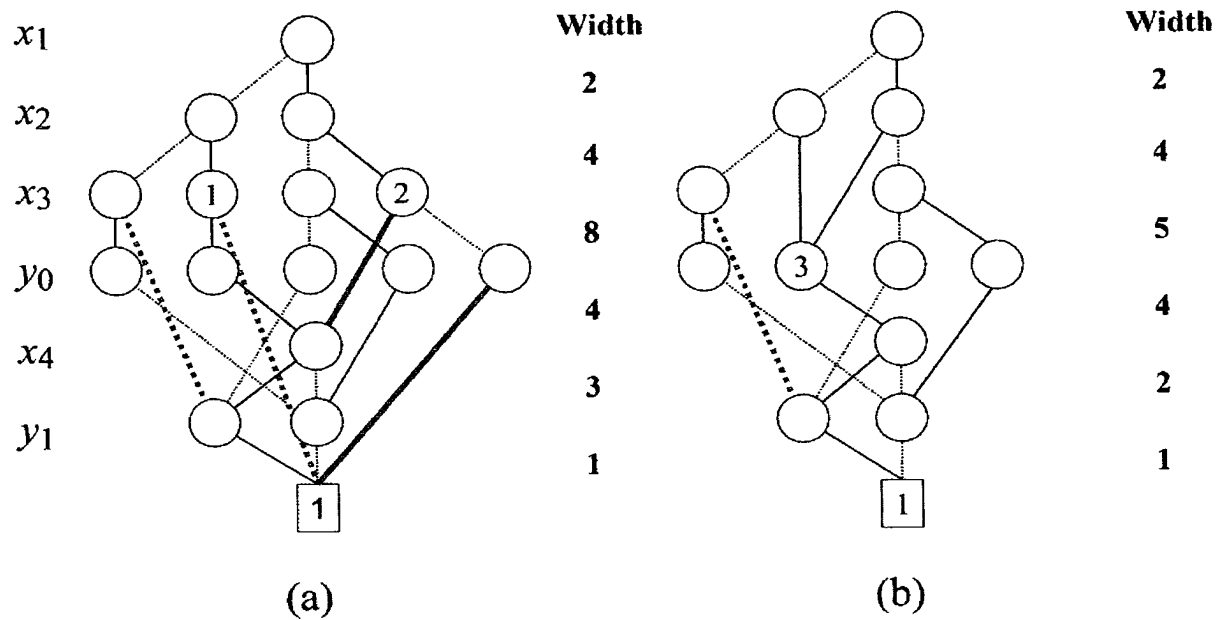
[図2]



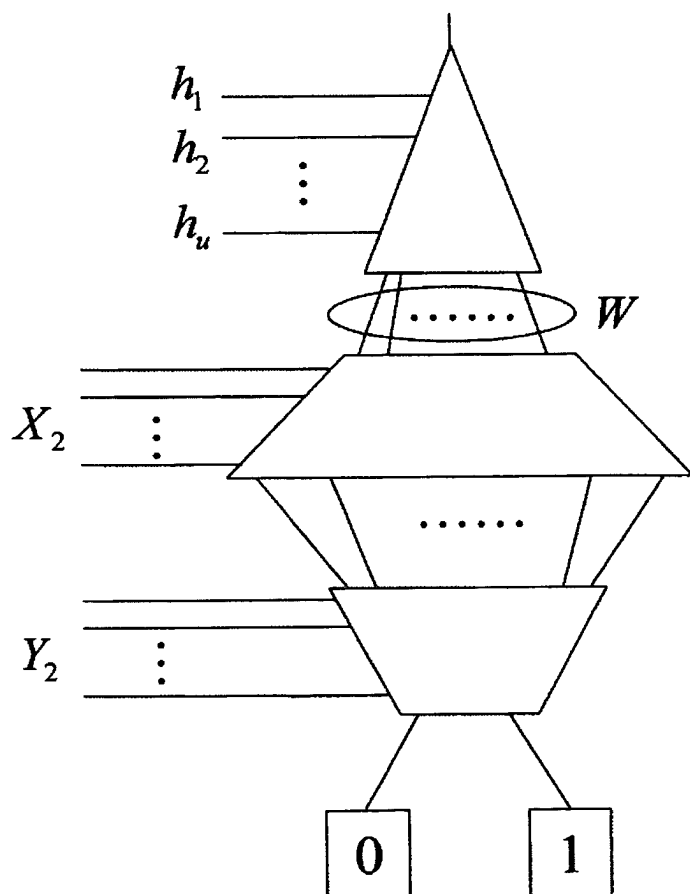
[図3]



[図4]



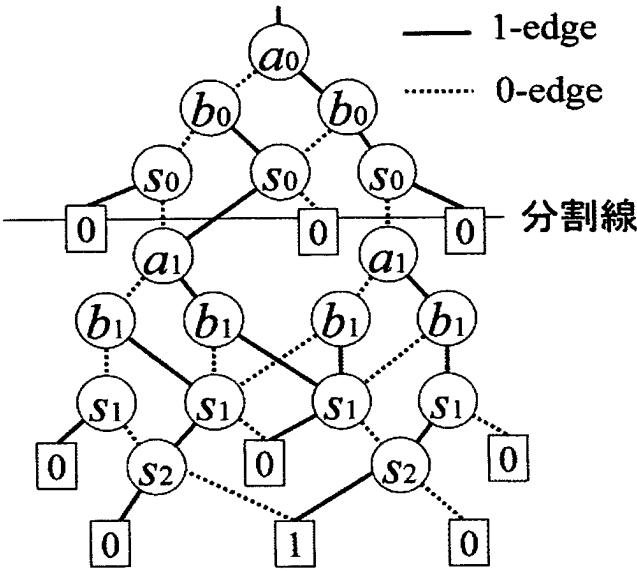
[図5]



[図6]

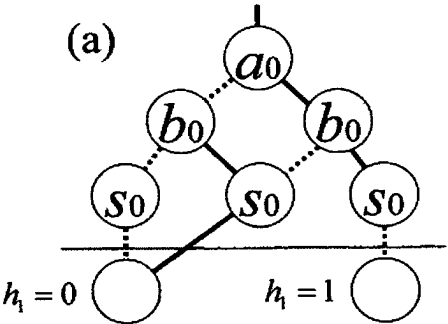
a_0	b_0	a_1	b_1	s_0	s_1	s_2
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
0	1	1	1	1	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	1

(a)

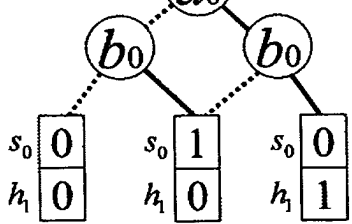


(b)

[図7]

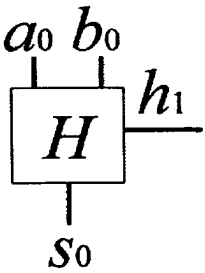


(b)

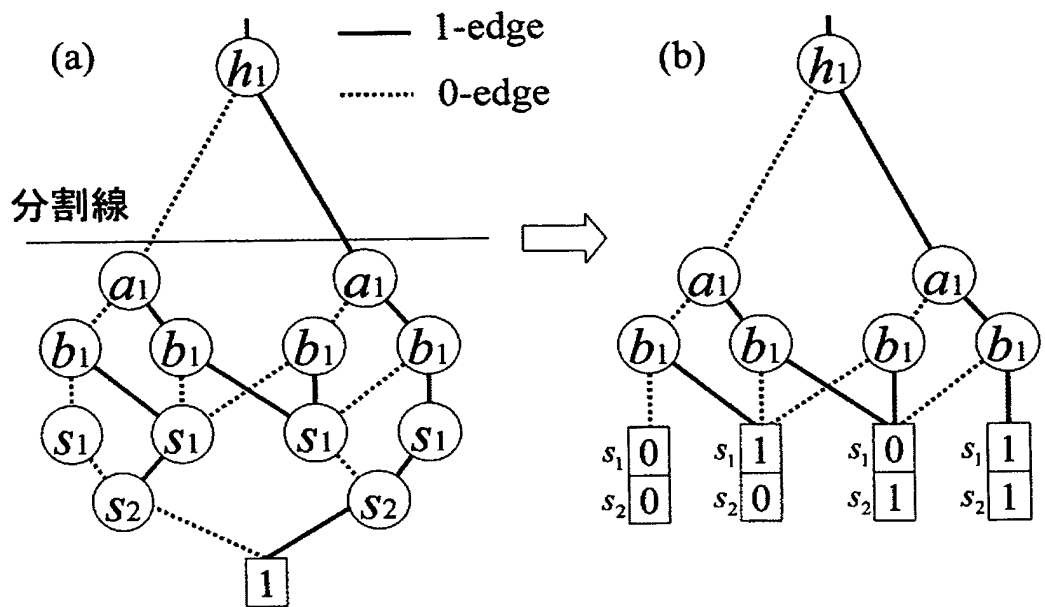


(c)

a_0	b_0	s_0	h_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



[図8]

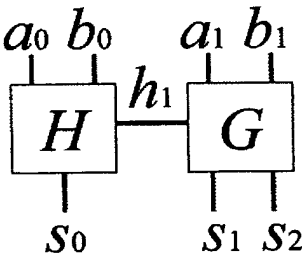


[図9]

(a)

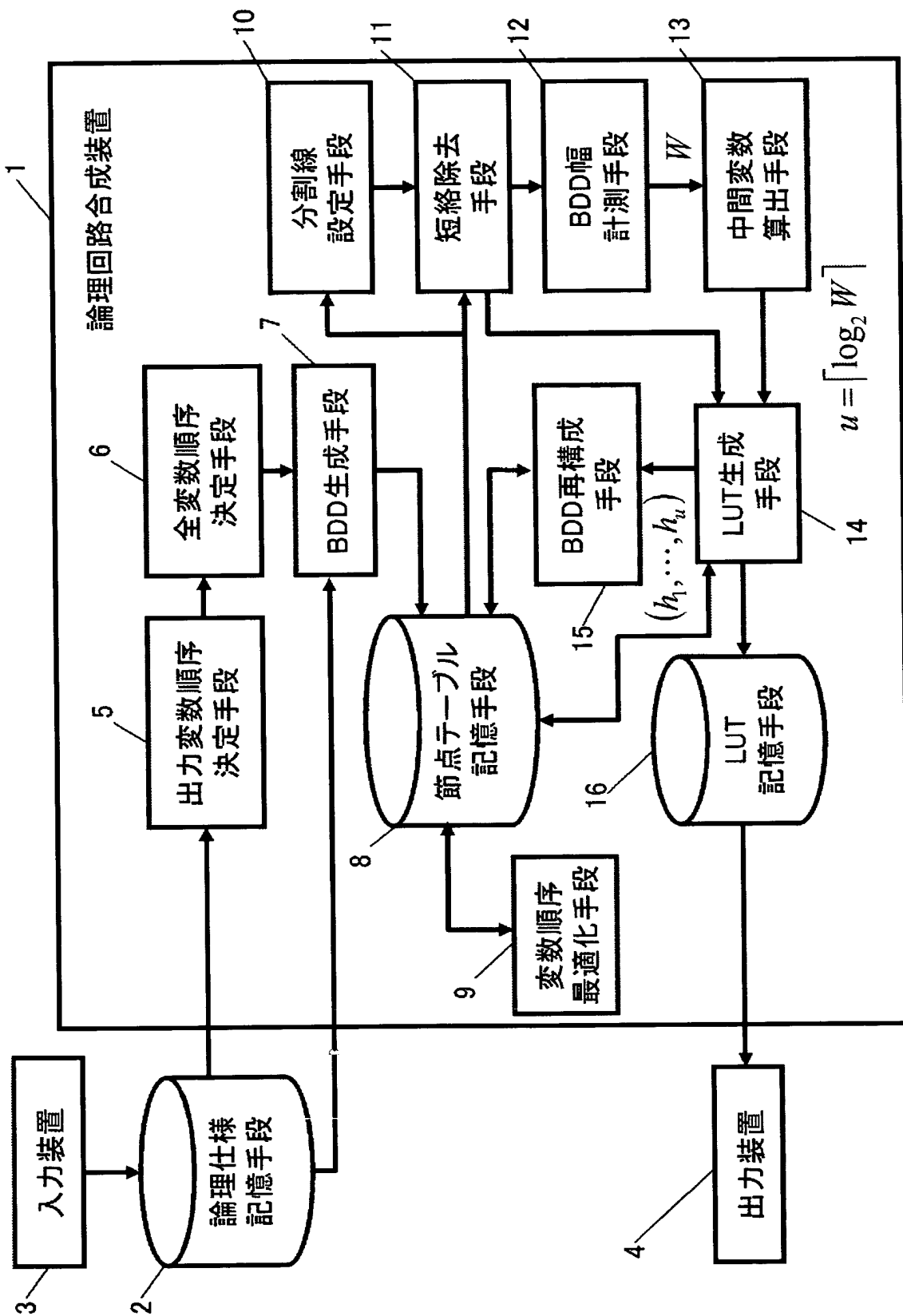
h_1	a_1	b_1	s_1	s_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b)

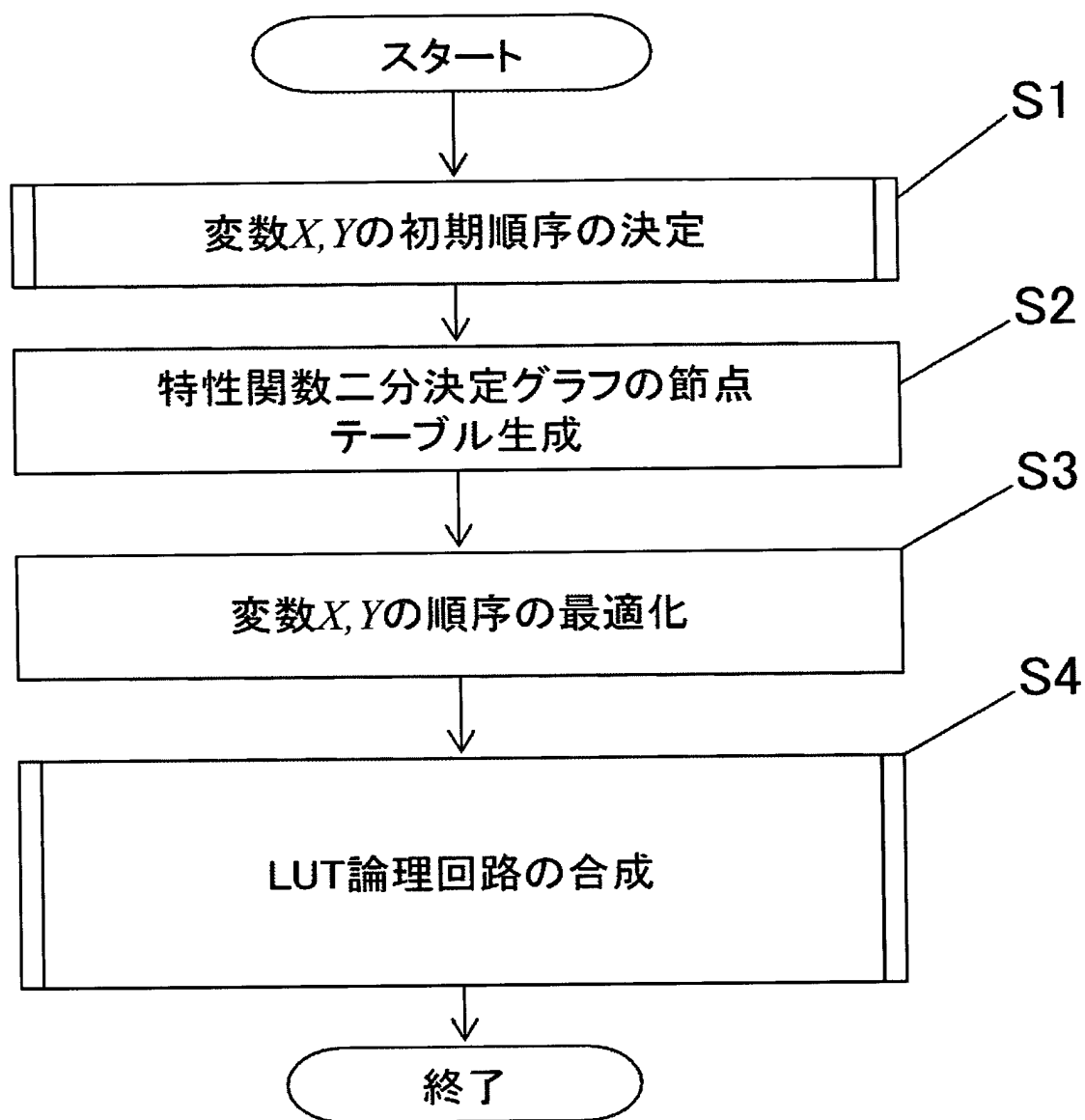


LUT数: 4
段数: 2

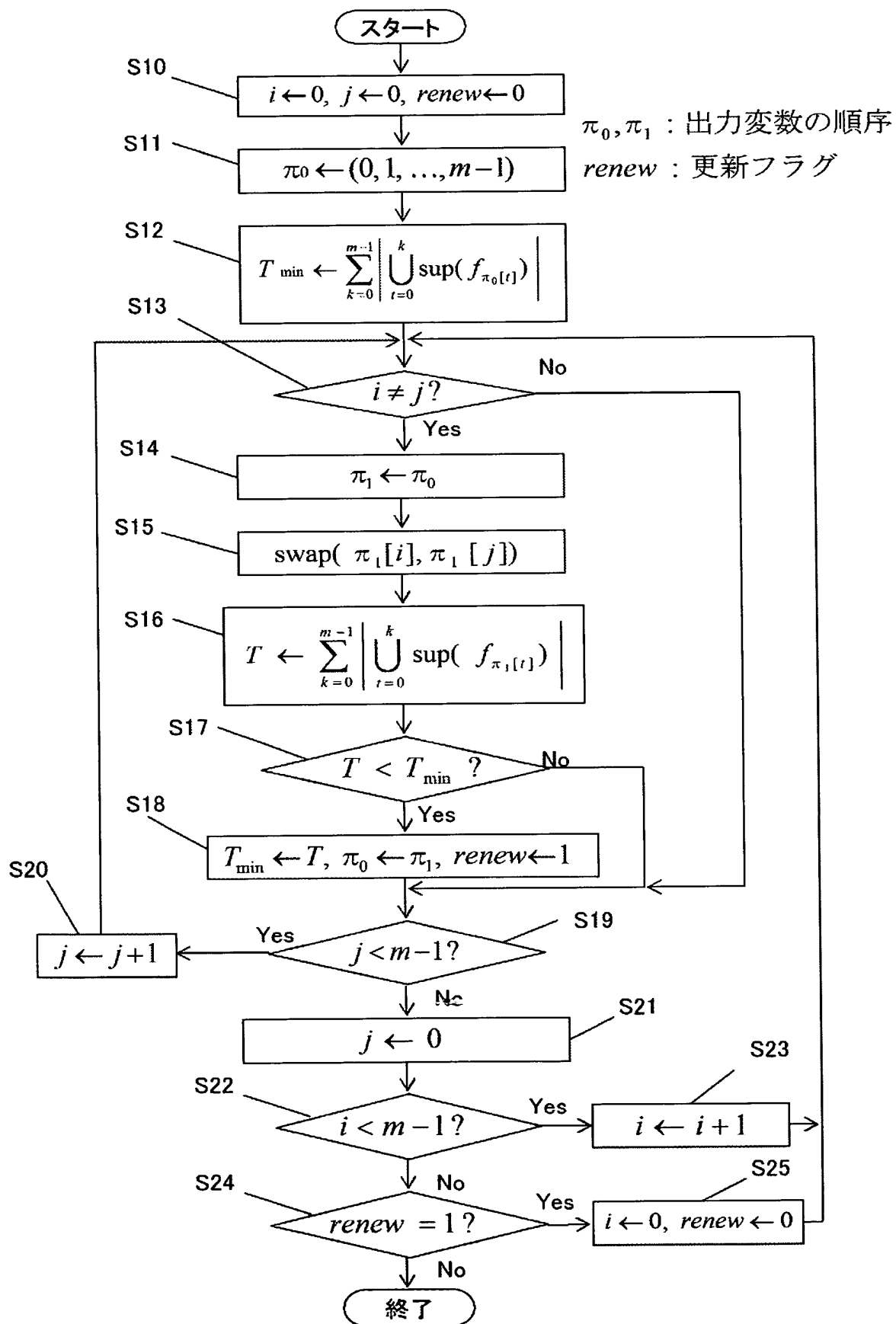
[図10]



[図11]

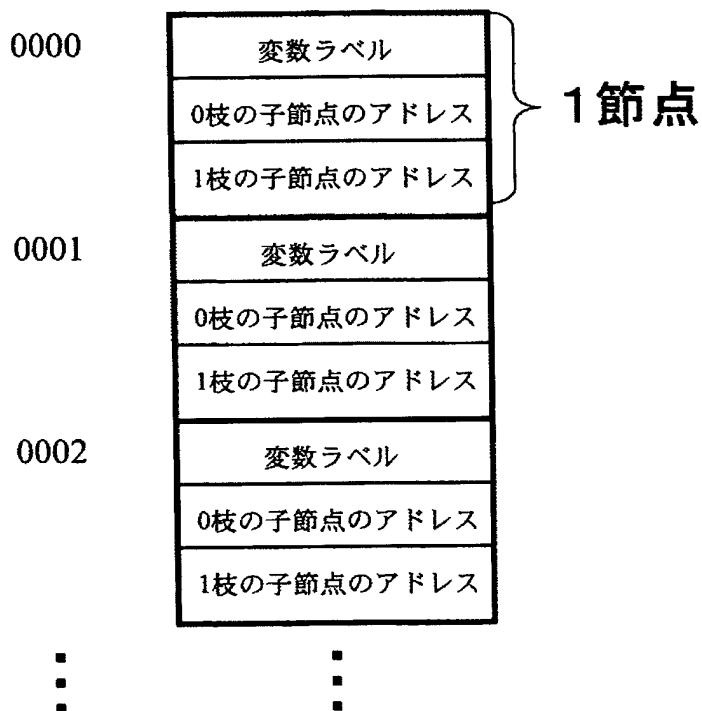


[図12]



[図13]

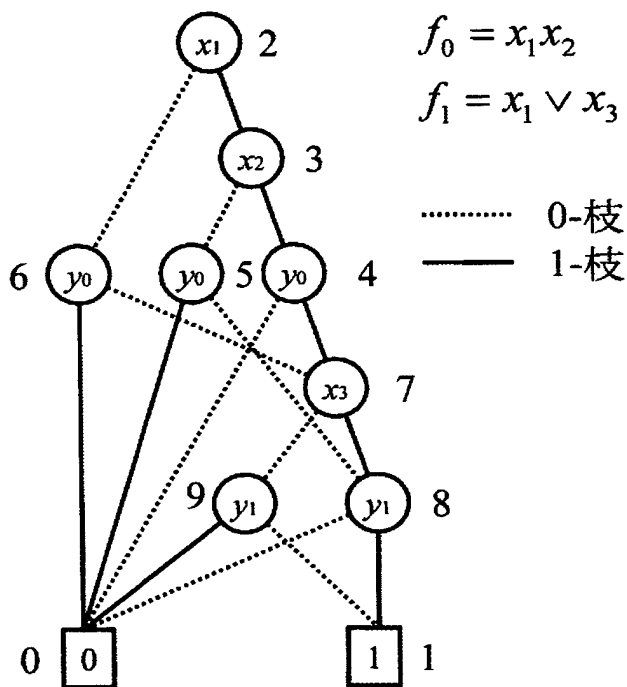
アドレス



[図14]

アドレス

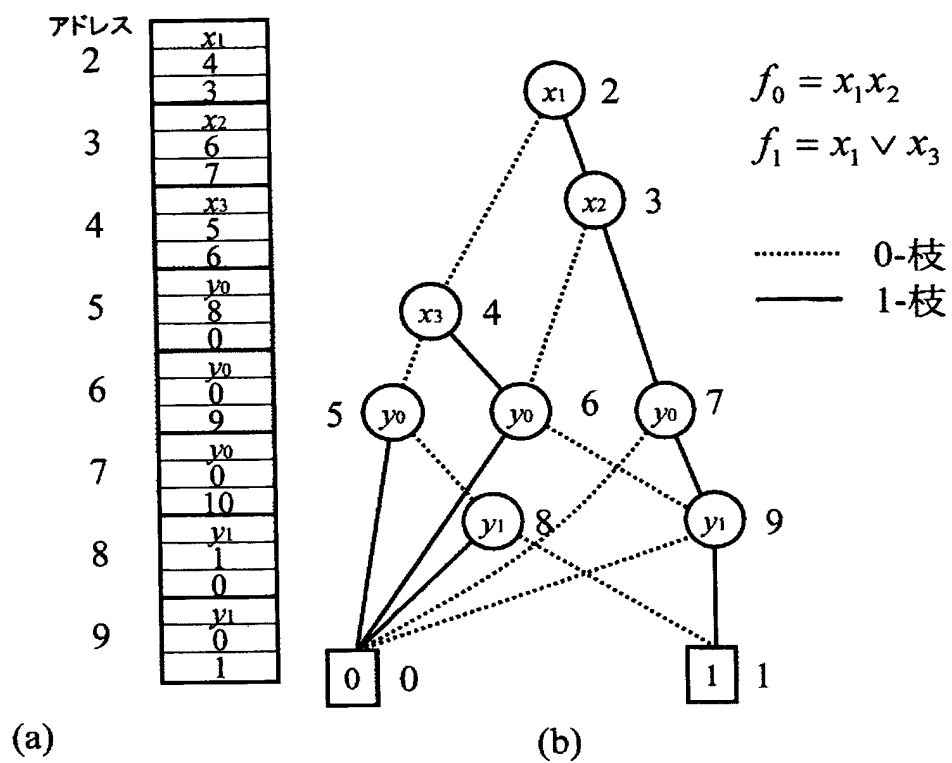
2	x_1
	6
	3
3	x_2
	5
	4
4	y_0
	0
	7
5	y_0
	8
	0
6	y_0
	7
	0
7	x_3
	9
	8
8	y_1
	0
	1
9	y_1
	1
	0



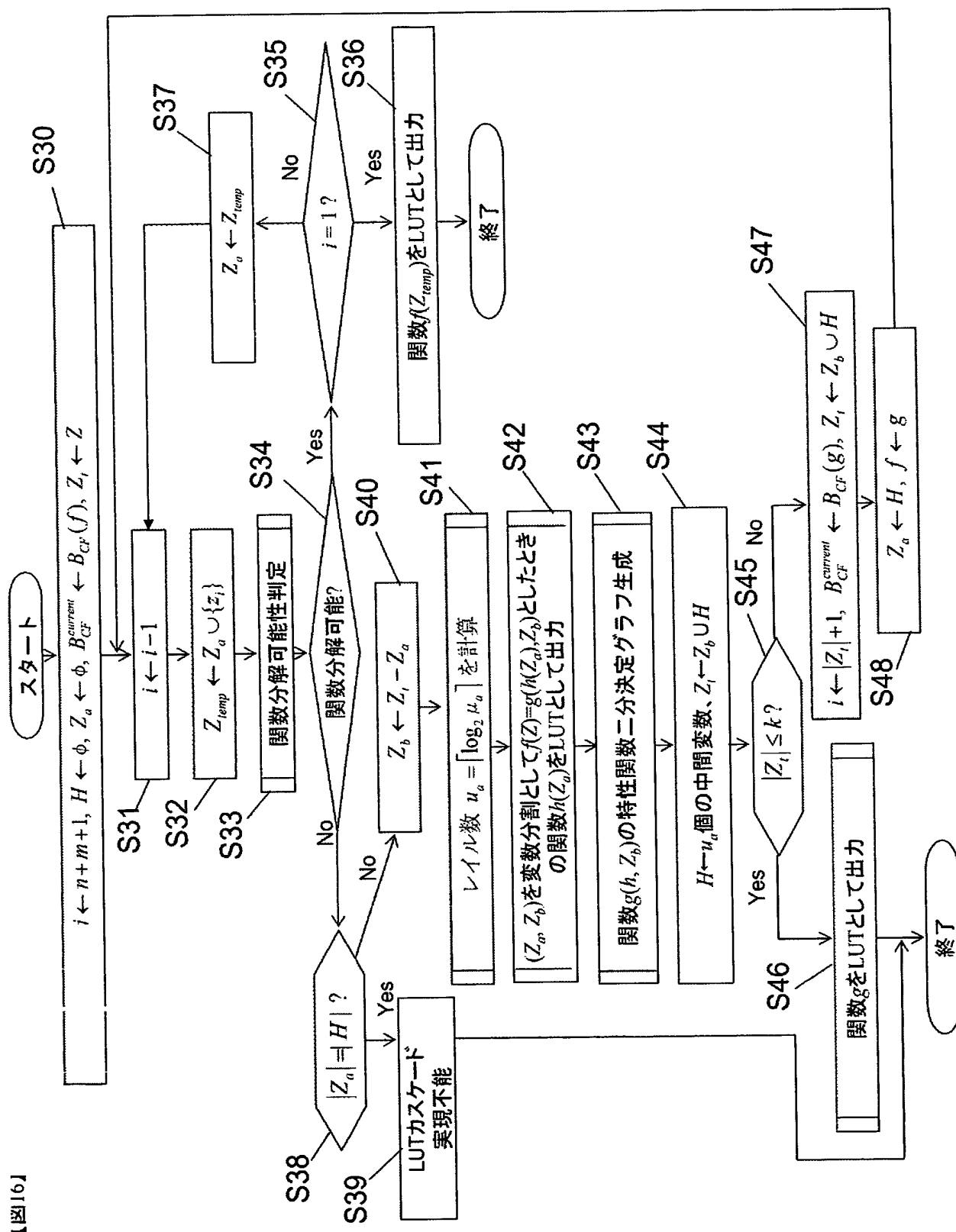
(a)

(b)

[図15]



[図16]

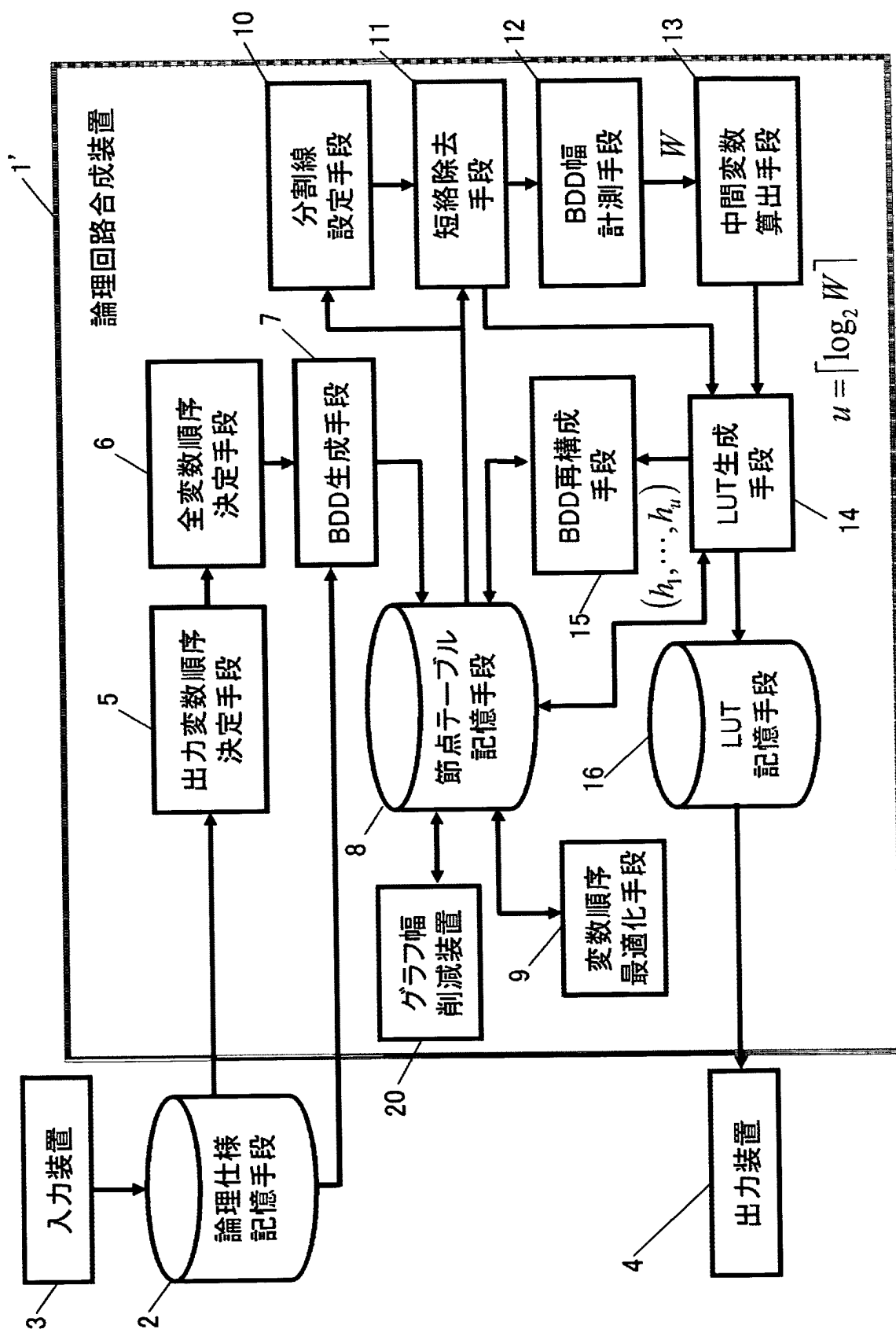


[図17]

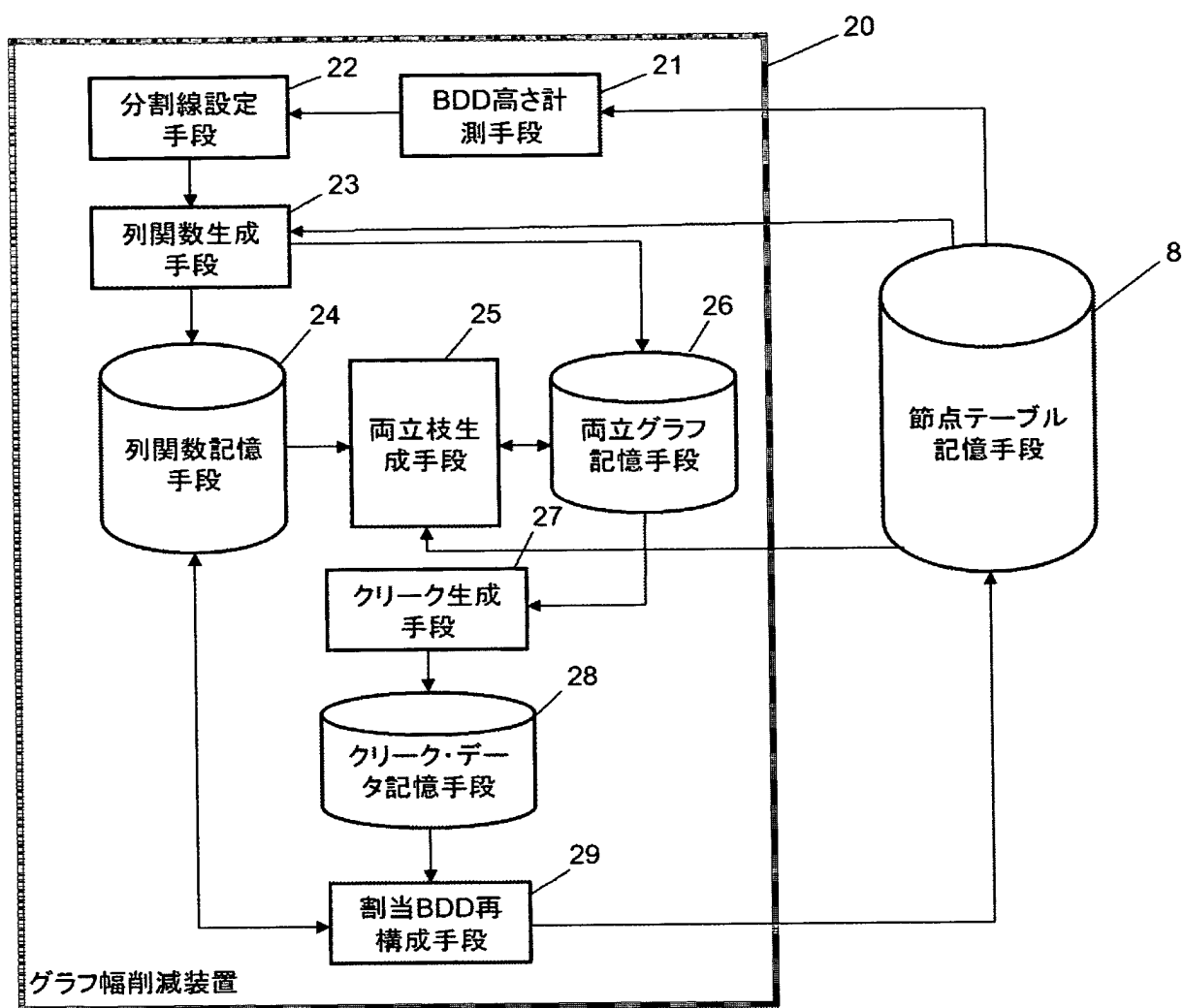
Name	In	Out	本発明の方法			Mishchenko ^[15]		
			LUT	段数	Cas	LUT	段数	Cas
C1908	33	25	320	11	5	3015	18	30
apex1	45	45	115	12	1	213	10	5
apex6	135	99	385	18	4	728	18	18
duke2	22	29	46	4	1	51	3	4
term1	34	10	50	8	1	37	6	2
ttt2	24	21	54	6	1	47	4	3

LUTの入力数を $k=10$ とした場合

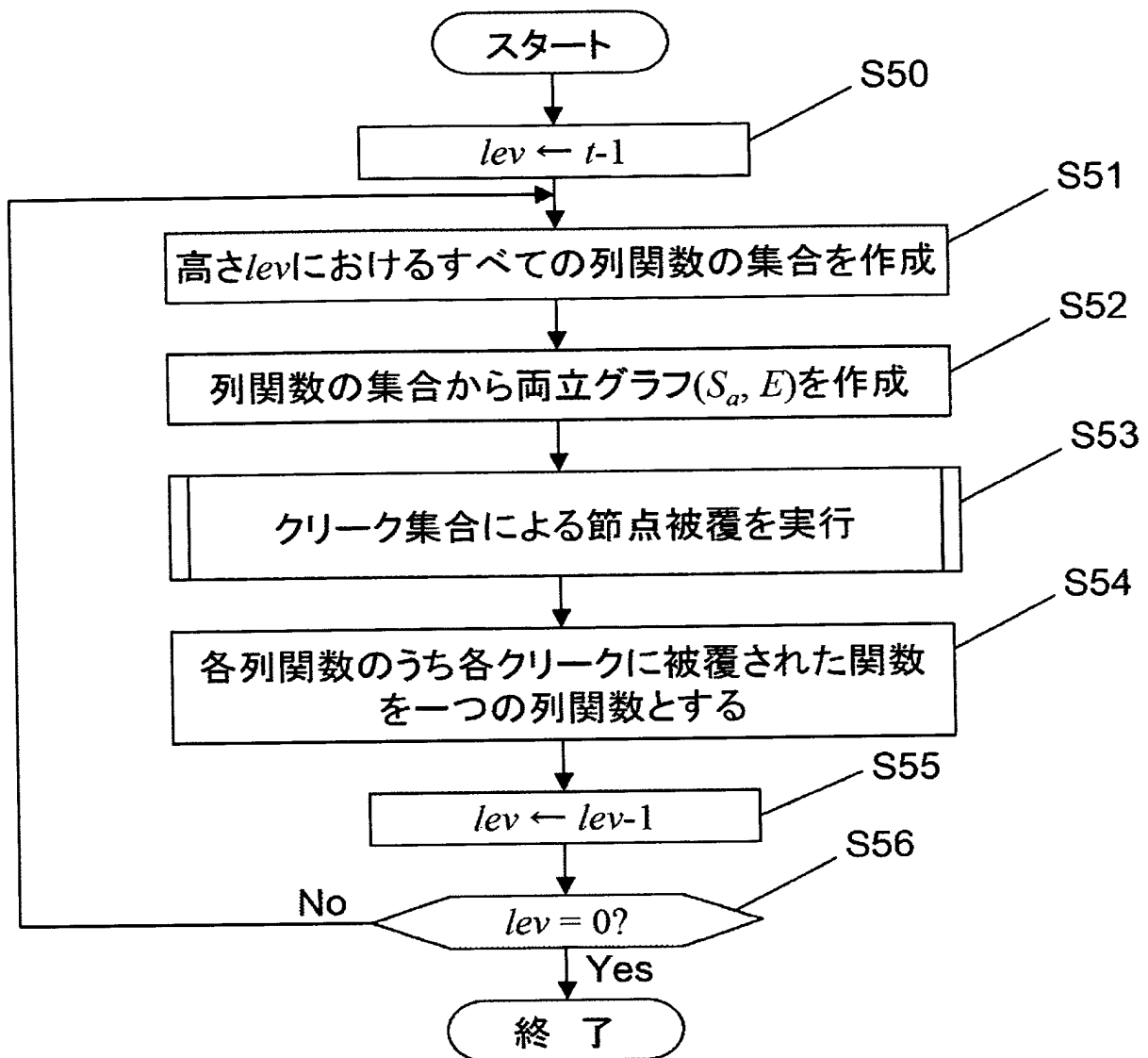
[図18]



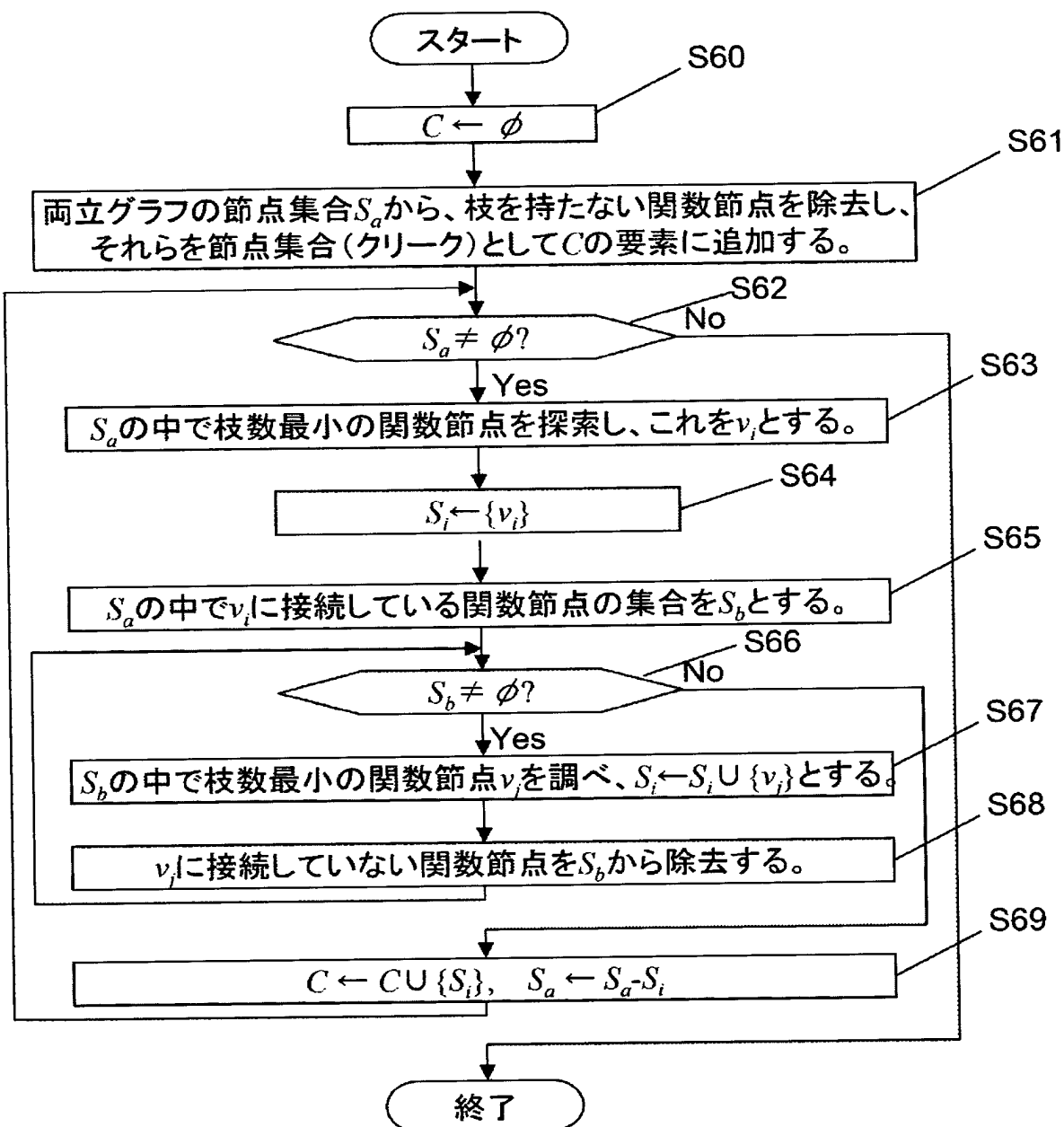
[図19]



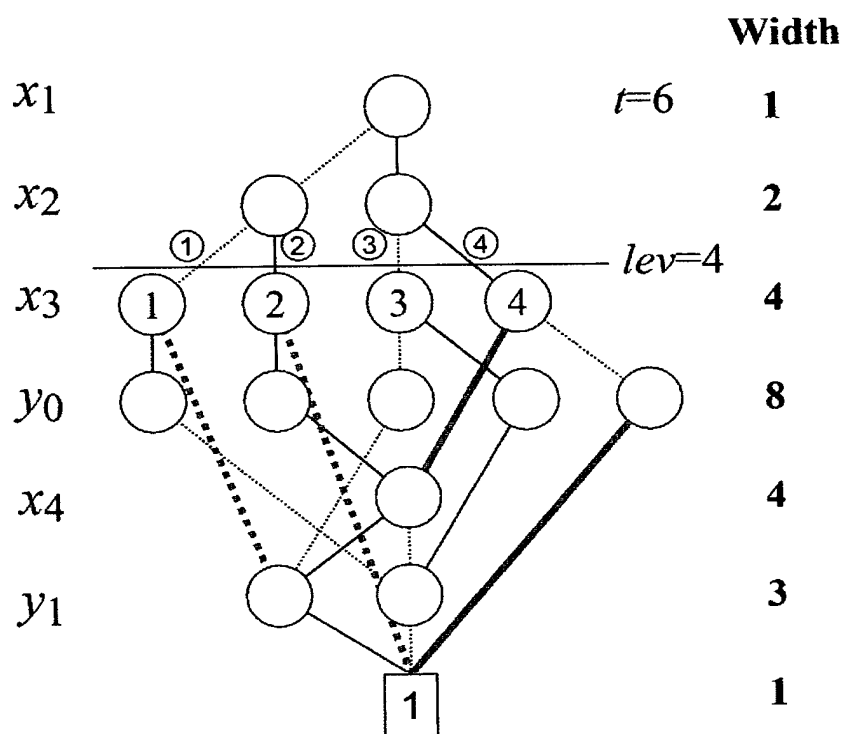
[図20]



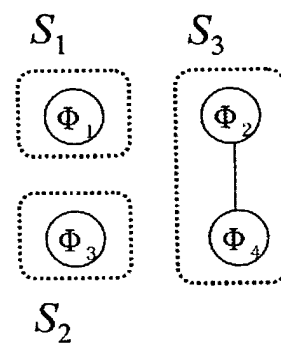
[図21]



[図22]

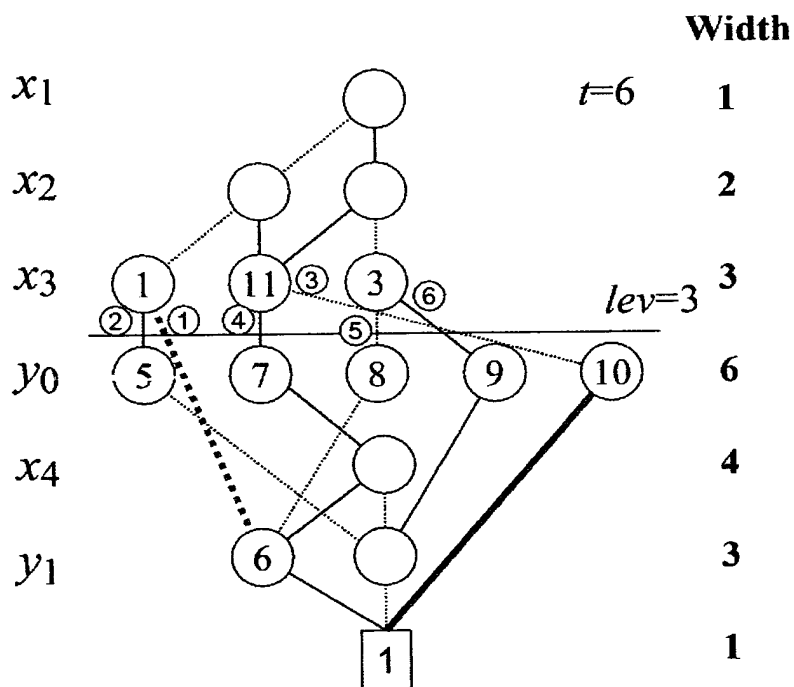


(a)

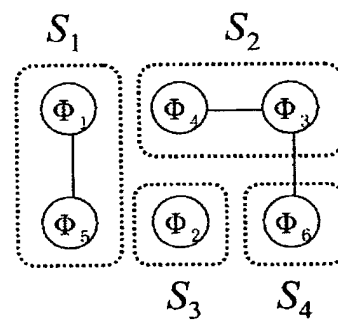


(b)

[図23]

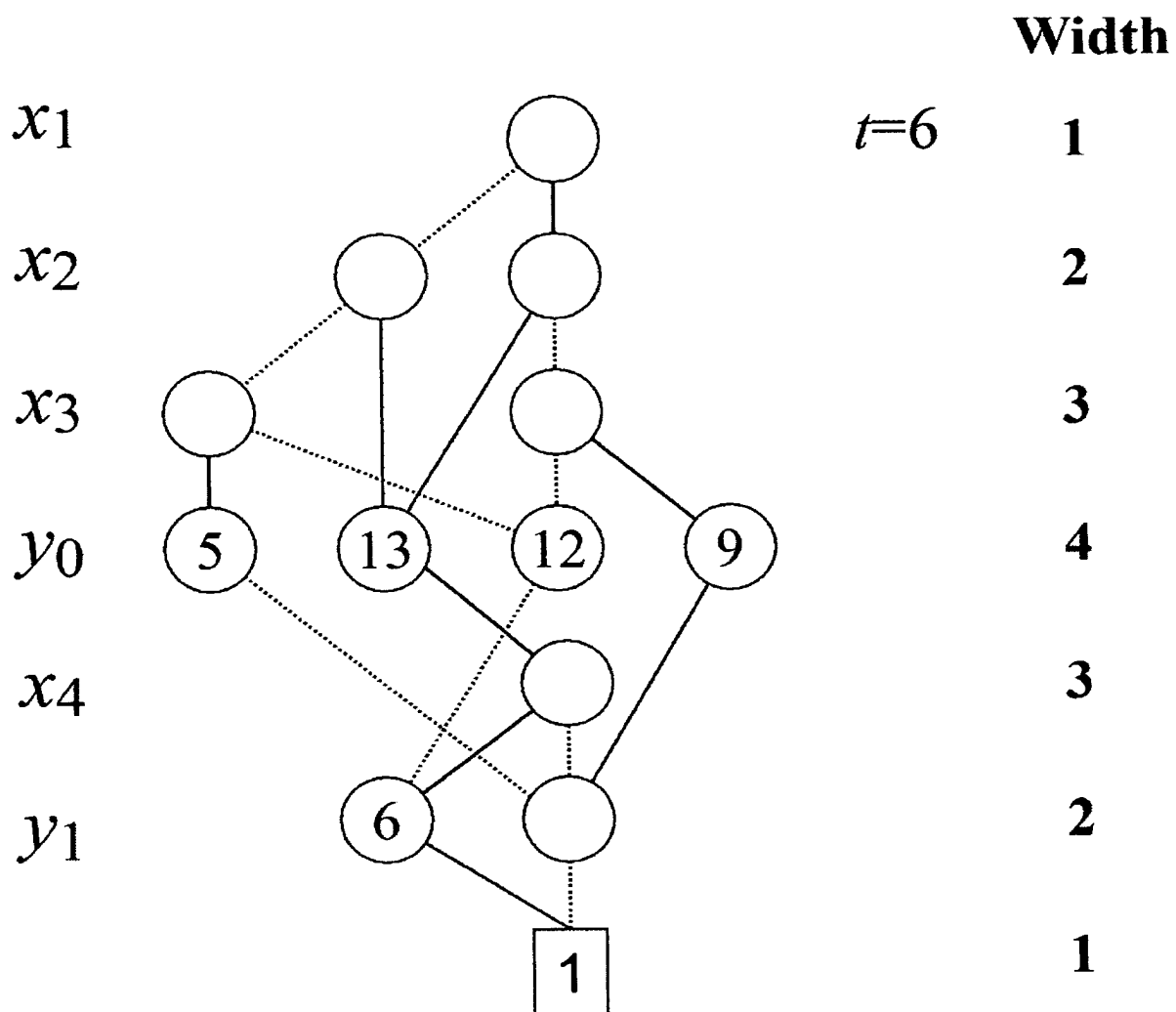


(a)

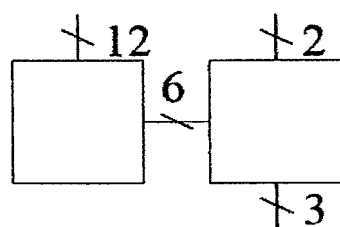
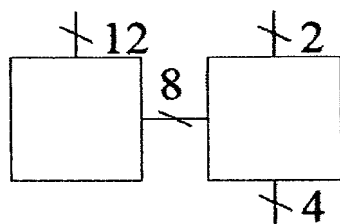
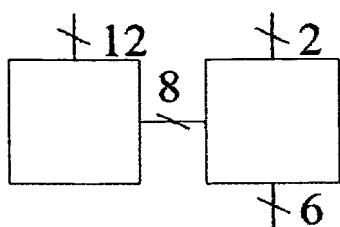


(b)

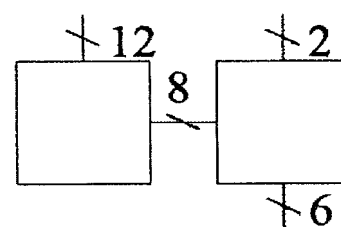
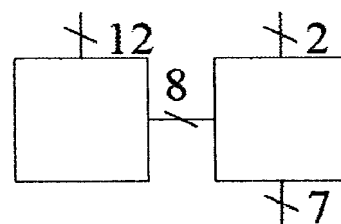
[図24]



[図25]



(a)



(b)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2004/017263

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl⁷ G06F17/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl⁷ G06F17/50

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
JSTPLUS FILE(JOIS), IEEE Explore

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	Y. IGUCHI et al., Realization of Mutiple-Output Functions by Reconfigurable Cascades, Proceedings of 2001 International Conference on Computer Design, 23 September, 2001 (23.09.01), Austin TX USA, pages 388 to 393	17 1-16
X A	Tsutomu SASAO et al., "Tashutsuryoku Kansu no Cascade Jitsugen to Saikosei Kano Hardware ni yoru Jitsugen", The Institute of Electronics, Information and Communication Engineers Gijutsu Kenkyu Hokoku, 06 April, 2001 (06.04.01), Vol.101, No.3 (FTS2001-1-13), pages 57 to 64	17 1-16

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
14 December, 2004 (14.12.04)

Date of mailing of the international search report
28 December, 2004 (28.12.04)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

A. 発明の属する分野の分類 (国際特許分類 (IPC))

Int. Cl⁷ G06F17/50

B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (IPC))

Int. Cl⁷ G06F17/50

最小限資料以外の資料で調査を行った分野に含まれるもの

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

JSTPLUSファイル (JOIS), IEEE Explore

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
X A	Y. Iguchi et al., Realization of Multiple-Output Functions by Reconfigurable Cascades, Proceedings of 2001 International Conference on Computer Design, 2001.09.23, Austin TX USA, p388-393	17 1-16
X A	笹尾勤ほか、多出力関数のカスケード実現と再構成可能ハードウェアによる実現、電子情報通信学会技術研究報告、2001.04.06、第101巻 第3号(FTS2001-1-13)、57-64頁	17 1-16

☐ C欄の続きにも文献が列挙されている。☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」 特に関連のある文献ではなく、一般的技術水準を示すもの
「E」 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)
「O」 口頭による開示、使用、展示等に言及する文献
「P」 国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」 国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」 同一パテントファミリー文献

国際調査を完了した日

14.12.2004

国際調査報告の発送日

28.12.2004

国際調査機関の名称及びあて先

日本国特許庁 (ISA/JP)

郵便番号100-8915

東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

田中 幸雄

5H

9191

電話番号 03-3581-1101 内線 3531

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☒ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.